Politechnika Poznańska
Wydział Elektryczny
Instytut Informatyki

Maciej Klemensowicz

System do przeprowadzania wzrokowych eksperymentów psychofizjologicznych z wykorzystaniem multisensora Jazz

Praca magisterska
wykonana pod kierunkiem
dr inż. Jacka Jelonka

Poznań, czerwiec 2003 r.

Serdeczne podziękowania dla

dr inż. Jacka Jelonka za inspirację
i pomoc w realizacji pracy

oraz prof. dr hab. inż. Jana Obera
za niezwykłą życzliwość i wielogodzinne konsultacje

Spis treści

Rozdział 1

Wstęp

1.1 Motywacje

Przyszło nam żyć w okresie intensywnego rozwoju mediów. Często mówi się, że żyjemy w czasach pisma obrazkowego. Zewsząd otaczają nas obrazy (w ten sposób podaje nam się codzienny zestaw najważniejszych informacji), chodzimy do kina (ruchomy obraz jest dla nas dobrą rozrywką), a fotografia została już dawno temu podniesiona do rangi sztuki. Obraz może być symboliczny, często służy komunikacji — jest swego rodzaju językiem. Ponadto czytanie jest niewątpliwie jedną z najważniejszych umiejętności cywilizacyjnych, dzięki której sprawnie i efektywnie funkcjonujemy we współczesnym społeczeństwie.

Zarówno czytanie jak i przyswajanie wiedzy z informacji obrazowej jest możliwe dzięki niezwykłemu aparatowi widzenia — narządowi wzroku. To, w jaki sposób posługujemy się wzrokiem na co dzień sprawia, że oczy są dla nas przede wszystkim urządzeniami wejścia, używając terminologii komputerowej. Dzięki nim, a także całemu przetwarzaniu odbywającemu w odpowiednich częściach naszego mózgu, potrafimy wzbogacić naszą wiedzę o świecie w oparciu o informacje pochodzące z „kanału wzrokowego".

Ze względu na uwarunkowania środowiskowe oraz specyficzną budowę ludzkiego oka w procesie ewolucji wykształcił się mechanizm poruszania oczami. Ruch naszych oczu możemy wykorzystać jako źródło informacji o zjawiskach intrapsychicznych. Okazuje się, że ruchy te nie są wcale przypadkowe lecz wykazują duży stopień uporządkowania. Charakterystyka ruchu, kierunek, prędkość, przyspieszenie, czas reakcji zależą nie tylko od wizualnych bodźców zewnętrznych, ale od sposobu i szybkości przetwarzania obserwowanych przez nasz umysł obrazów. Ruchy te są uwarunkowane także tym, co w danej chwili robimy, w jaki sposób pracuje nasz mózg, czy jesteśmy zmęczeni, pobudzeni, itd. Obserwacja ruchów oka może być pomocna w określeniu

preferencji człowieka, diagnozowaniu wielu dysfunkcji ośrodkowego układu nerwowego, a także może być podstawą eksperymentów psychologicznych badających procesy poznawcze. Więcej na temat samej natury ruchów oczu znajduje się w rozdziale 2.

Ze względu na specyfikę ruchów oka do ich obserwacji i analizy potrzebne są specjalistyczne środki techniczne. Urządzenie służące do rejestracji ruchu oczu powinno pozwalać na pomiar położenia ludzkiego oka z dużą częstotliwością próbkowania co uwarunkowane jest jest dynamiką samych ruchów. Równocześnie nie powinno ograniczać swobody działania ani powodować uczucie dyskomfortu u osoby badanej. W niniejszej pracy korzysta się z przyrządu o nazwie „Jazz". Został on zaprojektowany i zbudowany przez prof. Jana Obera i jego zespół z Instytutu Biocybernetyki i Inżynierii Biomedycznej Polskiej Akademii Nauk. Więcej o samym urządzeniu można dowiedzieć się z rozdziału 2.3.2.

Posiadając odpowiednie urządzenie pomiarowe i chcąc przeprowadzać wzrokowe eksperymenty psychofizjologiczne potrzebujemy dodatkowe narzędzie służące zarówno do przygotowywania eksperymentów okoruchowych jak również magazynowania i przetwarzania uzyskanych danych. W epoce gwałtownego rozwoju techniki cyfrowej naturalnym wyborem staje się komputer z dedykowanym do tego oprogramowaniem. Stworzenie takiego środowiska podyktowane zostało szerokim zastosowaniem ruchu oka w wielu dziedzinach: psychologii, psychiatrii, neurologii i wielu innych.

1.2 Cel i zakres pracy

Głównym celem niniejszej pracy jest stworzenie oprogramowania (nazwanego „UniJazz" — uniwersalny Jazz), które pozwala na budowanie i wykonywanie wzrokowych eksperymentów psychofizjologicznych z wykorzystaniem multisensora „Jazz". Umożliwi ono na zapoznanie się z podstawowymi właściwościami ruchów oka wszystkim osobom zainteresowanym (biologom, lekarzom, w tym okulistom, psychologom, psychiatrom, itd.). Do oprogramowania będzie dołączony zestaw podstawowych eksperymentów, a użytkownik będzie mógł je rozbudowywać, lub też samodzielnie tworzyć własne.

Najważniejsze cechy przygotowanego oprogramowania, a także wynikające z nich zadania szczegółowe:

- możliwość projektowania i wykonywania eksperymentów okoruchowych w oparciu o stworzony w tym celu dedykowany język skryptowy,
- łatwość w użytkowaniu (graficzny interfejs użytkownika), dostępność dla osób niezwiązanych bezpośrednio z informatyką,

- współpraca z urządzeniem pomiarowym (w tym przypadku z „Jazz'em"),
- możliwość składowania przebiegów ruchu oka na dysku w celu ich późniejszej analizy.

Więcej na temat funkcjonalności powstałego oprogramowania można dowiedzieć się z rozdziału 3, natomiast informacje na temat implementacji i współpracy z urządzeniem pomiarowym zawarte są w rozdziale 5.

Do oprogramowania dołączony jest zestaw eksperymentów, które pokazują możliwości stworzonego oprogramowania. Eksperymenty te zostały zaprojektowane w ramach niniejszej pracy przy dużym zaangażowaniu eksperta z dziedziny ruchu oka prof. Jana Obera. Każdy z tych eksperymentów obrazuje specyfikę określonych typów ruchu oka. W celu zweryfikowania systemu w praktyce diagnostycznej dodatkowo w ramach tej pracy przeprowadzono badania dotyczące występowania symptomów schizofrenii przejawiających się w ruchu oka. Odpowiednie ku temu celu eksperymenty badawcze zostały zaprojektowane i przeprowadzone z wykorzystaniem środowiska „UniJazz". Więcej Informacji na temat praktycznych zastosowań oraz wyników przeprowadzonych badań można znaleźć w rozdziale 6.

Rozdział 2

Ruchy oczu i ich pomiar

Narząd wzroku jest bardzo ważny dla funkcjonowania człowieka w świecie. Tak, wydawałoby się, proste zadanie jak czytanie tekstu, czy śledzenie uciekającego tramwaju wymaga już od naszych oczu całkiem skomplikowanego zestawu ruchów. Co więcej są one w ścisły sposób uporządkowane.

> „Ruchy oczu mają charakter wysoce uporządkowany i związane są zarówno z wykonywanym zadaniem, jak i z kolejnymi procesami analizy informacji."[10]

Stwierdzenie takie daje podstawy do tego, by naukowymi metodami badać nie tylko naturę samych ruchów, ale także (co jest chyba najciekawsze, a na pewno praktycznie użyteczne) ich powiązanie z naszym funkcjonowaniem umysłowym. Tutaj zresztą włączają się różne dzidziny nauki czyniąc całe zagadnienie interdyscyplinarnym. Najpierw okulistyka i biofizyka, jako że mamy do czynienia z fizjologią widzenia. Przede wszystkim jednak neuropsychologia, która dzięki swoim metodom eksperymentalnym umożliwia podpatrywanie i wyjaśnianie zjawisk związanych zarówno z ruchem oka jak również z procesami poznawczymi jemu towarzyszącymi. Umiejętność konstruowania odpowiednich testów daje prawdziwe możliwości badawcze. Tym samym istnieje potrzeba powstania oprogramowania wspomagającego ich tworzenie i przeprowadzanie. Takim właśnie oprogramowaniem jest „UniJazz".

2.1 Budowa oka i jego ruch

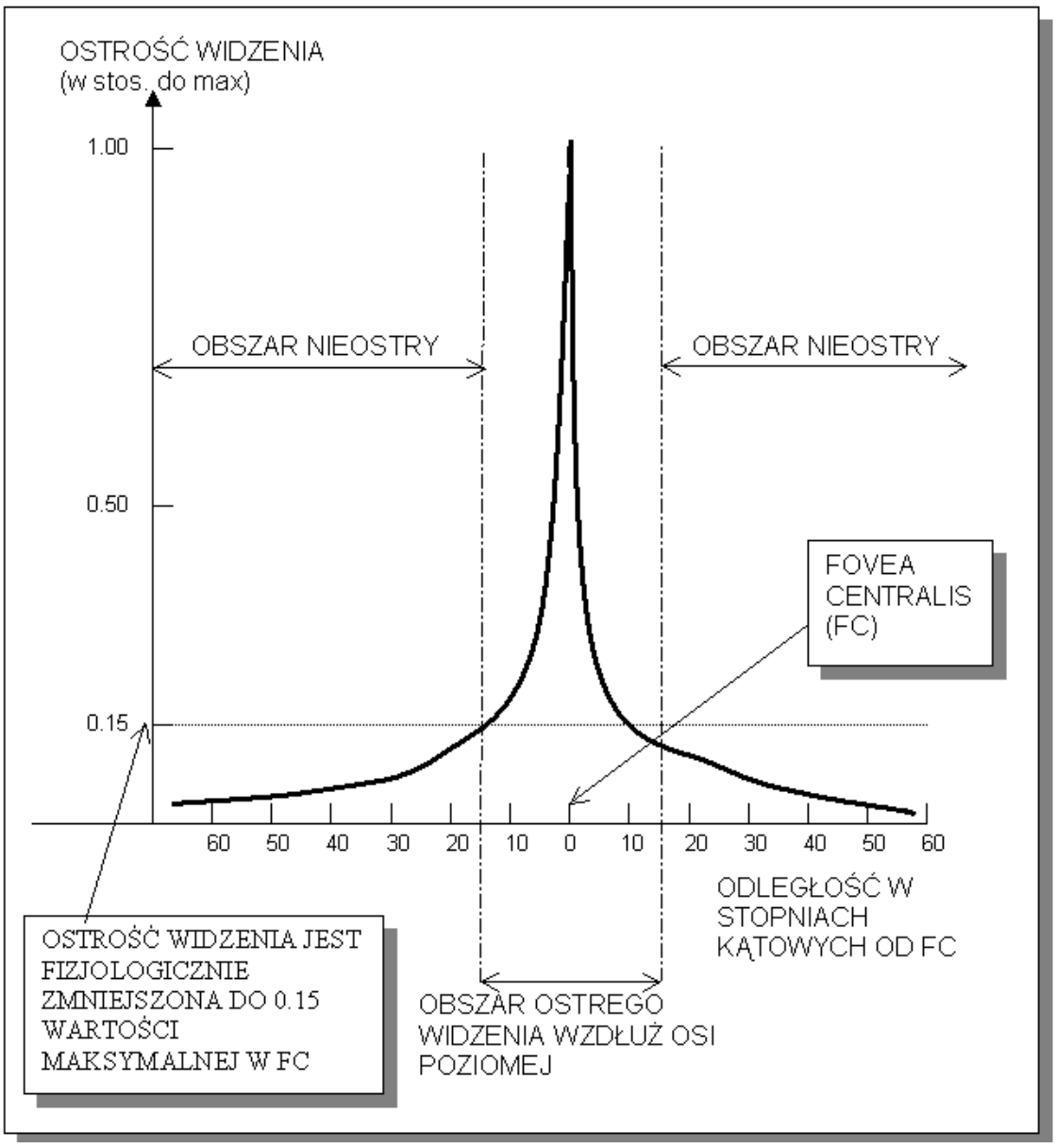
Ludzkie oko to wyspecjalizowany narząd służący do rejestrowania bodźców wzrokowych. Używając analogii związanej z budową aparatów fotograficznej oko jest połączeniem obiektywu (soczewki) oraz elementu rejestrującego obraz. Funkcję warstwy światłoczułej spełnia tutaj wyściółka komórek receptorowych (siatkówka), któ-

re wrażliwe są na trzy podstawowe barwy światła. Ich połączenie daje w rezultacie pełnobarwny obraz, który znamy z codziennych obserwacji.

Budowa i działanie samej warstwy rejestrującej obraz (siatkówki) jest efektem swoistego kompromisu pomiędzy jakością (rozdzielczością) obserwowanego obrazu a ilością danych niezbędnych do zakodowania informacji obrazowej. Nasze oko jest zdolne do ostrego widzenia wyłącznie w wąskim obszarze, wokół miejsca centralnego oka zwanego *fovea centralis* [2]. Tam właśnie ostrość widzenia osiąga wartość maksymalną, co spowodowane jest dużą gęstością komórek receptorowych oraz tym, że każda z nich połączona jest z mózgiem dedykowanym połączeniem nerwowym. Poza tymże obszarem ostrość widzenia spada gwałtownie, by w odległości około $\pm 15^\circ$ od miejsca centralnego osiągnąć wartość zaledwie 0,15 maksimum. Przybliżony profil ostrości widzenia jest przedstawiony na rysunku 2.1.

Surowa informacja obrazowa (pobudzenia komórek wzrokowych) musi być przesłana do odpowiednich partii mózgu zajmujących się przetwarzaniem sygnału. Natura wykorzystuje w tym celu nerw wzrokowy, będący skupieniem wielu włókien nerwowych. Ich duża liczba wynika z tego, że informacja obrazowa wymaga dużej pojemności kanału przesyłowego. Gdybyśmy jednak wyobrazili sobie siatkówkę oka, na której komórki receptorowe są rozłożone równomiernie na całej jej powierzchni, przy tym ich gęstość byłaby porównywalna z tą występującą w *fovea centralis*, to nerw wzrokowy musiałby być grubości ludzkiego przedramienia. Dzięki zatem nierównomiernemu rozmieszczeniu komórek światłoczułych na siatkówce, a tym samym zróżnicowanej rozdzielczości ludzkiego oka na całej jego powierzchni nasz mózg potrafi „poradzić sobie" z dużą ilością przesyłanych danych.

Wiemy już zatem, że ludzkie oko potrafi rejestrować obrazy z dużą rozdzielczością w niewielkim obszarze w środku pola widzenias oraz ze stosunkowo niską na peryferiach. Aby jednak móc dostrzegać wyraźne szczegóły w każdym obszarze obserwowanej przez nas sceny musi istnieć mechanizm naprowadzający wybrany przez nas fragment pola widzenia na centralną część siatkówki. Najprostszą metodą jest obrócenie głowy. Nie jest to sposób doskonały, gdyż głowa ze względu na duży ciężar charakteryzuje się znaczną bezwładnością. To z kolei sprawia, że ruchy głowy są raczej powolne, ociężałe, a te gwałtowne prowadzą do szybkiego zmęczenia mięśni i w konsekwencji do uczucia bólu. Natura zaopatrzyła nasz gatunek (i nie tylko) w mechanizm ruchu gałek ocznych. Rozwiązanie wydaje się być bliskie perfekcji, jako że tym razem wprawiana w ruch jest stosunkowo niewielka masa. Oko waży zaledwie około 7 gramów [6]. To umożliwia ruchy z dużymi prędkościami i przyspieszeniami kątowymi.



Rysunek 2.1: Rozkład względnej ostrości widzenia

2.2 Typy ruchów oczu

Obserwując ruch oczu w różnych sytuacjach eksperymentalnych można wyodrębnić kilka typów ruchu. Każdy z nich związany jest z nieco inną funkcją. Wśród nich warto wyróżnić następujące:

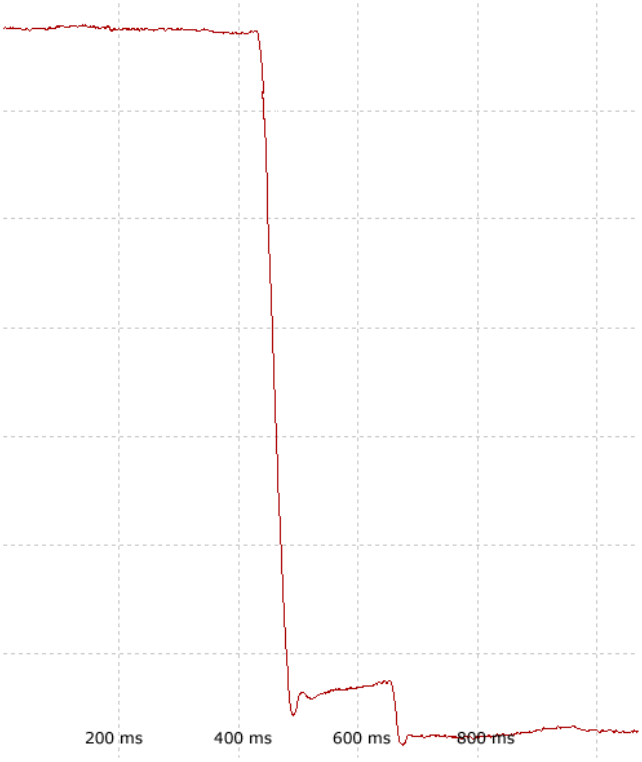
- śledzenie poruszających się obiektów, np. wodzenie wzrokiem za lecącym ptakiem,
- śledzenie poruszającej się sceny, np. obserwacja „uciekającego" krajobrazu z okna poruszającego się pociągu,
- zmiana punktu fiksacji, tj. szybkie przeniesienie wzroku np. z twarzy jednego rozmówcy na twarz drugiego.

Przy bliższym poznaniu jednak to co nazywamy „ruchem oka" jest sekwencją wyłącznie dwóch podstawowych ich rodzajów: ruchów sakkadycznych, inaczej szybkich (ang. *saccade*) i ruchów wolnych (ang. *slow eye movement*). Nazwy sugerują ich naturę, jednak po latach doświadczeń i obserwacji wydają się być nieco mylące, jako że najnowsze badania pokazują, że niektóre ruchy „wolne" mogą być szybsze od tych „szybkich" w pewnych specyficznych warunkach. Krótka charakterystyka obu typów przedstawiona poniżej pozwoli jednak na ich jednoznaczne rozróżnienie.

2.2.1 Ruchy sakkadyczne

Ruchy sakkadyczne (albo krótko sakkady) charakteryzują się zwykle wysoką prędkością i zawsze dużym przyspieszeniem kątowym. Są to gwałtowne przerzucenia wzroku z jednego punktu na inny. Podczas wykonywania ruchu gałki ocznej widzenie jest zablokowane aby przeciwdziałać powstawaniu rozmytego, niewyraźnego obrazu. Aby człowiek nie pozostawał „ślepy" przez dłuższy okres czasu (mogłoby to dla niego być niebezpieczne) sam ruch trwa kilkadziesiąt milisekund (typowo około $40 \div 50$ ms), prędkość kątowa sięga $900°$ na sekundę. Nie jesteśmy w stanie zauważyć zjawisk trwających tak krótko, dlatego nie odczuwamy bezpośrednio sakkady, a jedynie jej skutki — przerzucenie wzroku.

Sakkady są wykonywane niemalże cały czas, cyklicznie co kilkaset milisekund. Cały proces tzw. programowania sakkady i samo jego wykonanie trwa z reguły około 250 ms. Taki czas właśnie upływa od momentu wystąpienia bodźca, bądź woli człowieka powodującego przerzucenie wzroku, do momentu, kiedy oko znajdzie się w wymaganej pozycji.



Rysunek 2.2: Przykład ruchu sakkadycznego oka. Widoczna sakkada korekcyjna

Chociaż wszystkie sakkady mają wyżej wymienione wspólne cechy, to jednak poszczególne różnią się w szczegółach, w zależności od pełnionej funkcji. Poniżej przedstawiam krótką charakterystykę ważniejszych typów sakkad.

1. **Refiksacja** — sakkada wywołana pojawieniem się bodźca w innej niż centralnej części pola widzenia. Naturalnym odruchem w takiej sytuacji jest „zerknięcie" w tamtym kierunku, jednak sakkady refiksacyjne mogą być także wywołane aktem woli człowieka — spojrzenia w inna stronę. Bywa, że trudno jest powstrzymać się od odruchu refiksacyjnego, szczególnie wśród osób cierpiących na pewnego rodzaju dysfunkcje, np. chorobę Alzcheimera. Sakkady refiksacyjne można nazwać klasycznymi, jako że one pojawiają się najczęściej i występują w roli „modelowych". Czas trwania pełnego cyklu sakkady wynosi około 250 ÷ 300 ms.

2. **Sakkada korekcyjna** — występuje w połączeniu z refiksacją. Często zdarza się, że refiksacja jest niedokładna, co oznacza, że oko wykonuje niedokładny ruch sakkadyczny. W takiej sytuacji po około 160 ms następuje korekcja i pojawia się dodatkowa sakkada o znacznie mniejszej amplitudzie niż refiksacyjna. Ponieważ cały cykl trwa poniżej 200 ms jest całkowicie niezauważalny dla człowieka.

3. **Sakkada wielostopniowa** — to właściwie sakkada refiksacyjna złożona z kilku (2, rzadziej z 3) sakkad występujących po sobie w czasie krótszym niż 160 ms. Występują spontanicznie, statystycznie rzadziej niż pojedyncza sakkada refiksacyjna. Obserwuje się natomiast podwyższone prawdopodobieństwo ich występowania u osób chorych na parkinsonizm. Ich wielostopniowość jest całkowicie niezauważalna dla ludzkiej percepcji. Prawdopodobną przyczyną ich występowania jest czas niezbędny do uaktywnienia kolejnych włókien mięśniowych.

4. **Sakkada kompensacyjna** — wykonywana jest w celu skompensowania mikrodryftów powstających w trakcie trwania długiej fiksacji. Sakkada taka nosi też nazwę „mikrosakkady" ze względu na niewielką amplitudę ruchu. W trakcie trwania fiksacji powstają mikroruchy, które powodują, że z czasem obiekt zaczyna przesuwać się w kierunku peryferyjnego obszaru widzenia, a tym samym istnieje niebezpieczeństwo, że stanie się nieostry. Co pewien czas wykonywana jest zatem sakkada kompensacyjna, która „poprawia" dokładność fiksacji.

5. **Sakkada goniąca** — występuje w połączeniu z ruchem wolnym oka, głównie w pierwszej jego fazie. Jak sama jej nazwa wskazuje jej zadaniem jest „nadgonienie" (ang. *catch–up saccade*) za poruszającym się bodźcem.

6. **Sakkada ekspresowa** — charakteryzująca się krótkim czasem upływającym od pojawienia się bodźca do zakończenia trwania ruchu. Tego typu sakkady pojawiają się bardzo rzadko, można jednak prawdopodobieństwo ich wystąpienia wydatnie zwiększyć w specyficznych warunkach eksperymentalnych. Czas pełnego cyklu sakkady to około 150 ms. Ich występowanie można wiązać z tym, że zwalnianie zasobów ludzkiej uwagi wiąże się najprawdopodobniej z pewnym narzutem czasowym. Według poglądu wielu naukowców ludzka uwaga może być traktowana jako zasób, który można przydzielić tylko do jednej konkretnej czynności. Jeśli uda się w jakiś sposób uwolnić uwagę (odebrać ten zasób wykonywanej czynności, na przykład wpatrywaniu się w punkt) przed pokazaniem bodźca refiksacyjnego, wtedy wzrasta prawdopodobieństwo wystąpienia sakkady ekspresowej.

2.2.2 Ruchy wolne

Ruchy wolne, czasami nazywane „nadążnymi" mają na celu stabilizację obrazu na siatkówce oka. To dzięki nim możemy „podążać" wzrokiem za wybranym, poruszającym się obiektem, a także obserwować poruszającą się scenę. Tak jak ich nazwa

sugeruje, prędkości kątowe osiągane w większości codziennych sytuacji nie przekraczają 100°na sekundę. Można jednak stworzyć takie warunki laboratoryjne, w których uzyskuje się prędkości przewyższające 900°na sekundę. Mogą być więc szybsze od ruchów sakkadycznych. To co je od nich odróżnia, to nie tak wielkie przyspieszenie kątowe jak w przypadku sakkad.
Najważniejsze funkcje i typy ruchów wolnych przedstawia poniższa klasyfikacja:

1. **Śledzenie poruszającego się obiektu (ang. *smooth pursuit* albo *tracking*)** — pozwalają na wodzenie wzrokiem za obiektem. Początkowa faza ruchu, a także faza, w której ruch obiektu zmienia swoją charakterystykę (np. kierunek, prędkość) to zestaw sakkad (zwanych goniącymi).

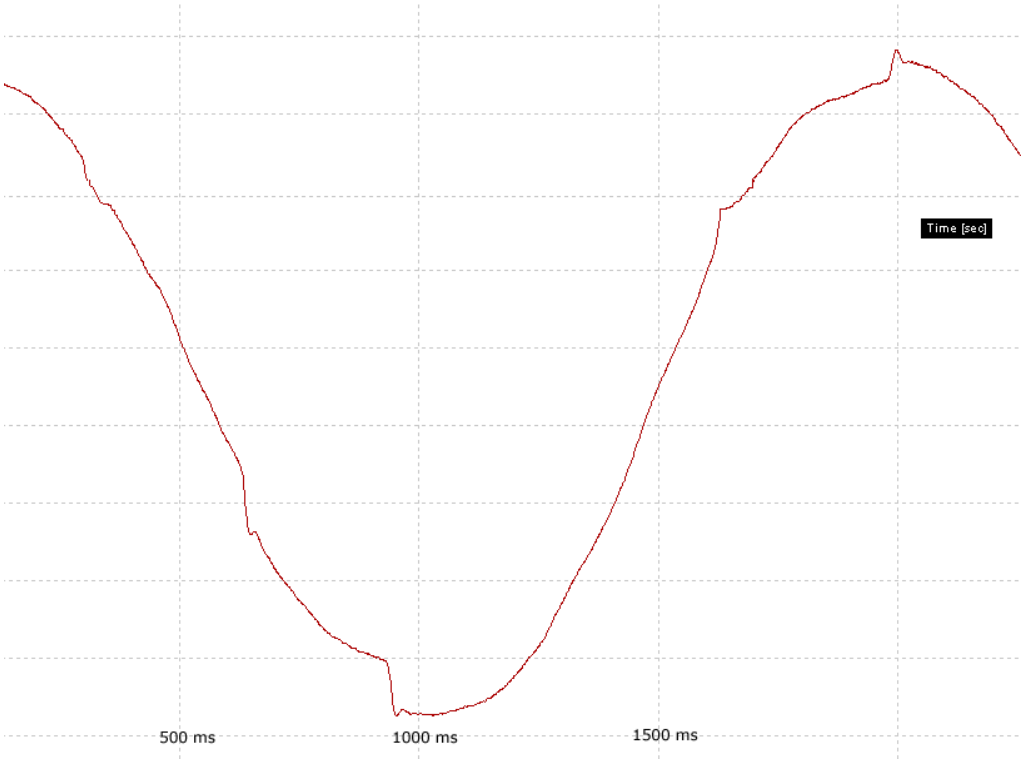
2. **Śledzenie poruszającej się sceny (ang. *optokinetic response*)** — umożliwia obserwację poruszającej się sceny, związany jest z występowanie reakcji oczopląsu. Efekt taki występuje na przykład podczas obserwacji „uciekającego" krajobrazu z okna jadącego pociągu.

3. **Stabilizacja obrazu podczas ruchu głową (ang. *okulomotor response*)** — pozwala utrzymać nieruchomy obraz na siatkówce oka kiedy poruszamy głową.

Ruchy wolne oczu *smooth pursuit* są wykonywane przez układ niezależny od sakkadycznego. Działa on podobnie do reprogramowalnego generatora funkcji ruchu. Raz zaprogramowana funkcja (będąca *de facto* kombinacją trajektorii i prędkości) nie wymaga oddzielnej, stałej kontroli, chyba że nastąpi zmiana w charakterystyce ruchu śledzonego obiektu. Jeżeli zdarzy się taka sytuacja, „generator ruchu" jest reprogramowany, czyli parametry ruchu oka są dostosowywane do aktualnych warunków. U osób chorych na schizofrenię obserwuje się wydłużony czas reprogramowania tego układu, więcej na ten temat i przeprowadzone badania znajdują się w rozdziale 6.

2.3 Metody pomiarowe

Obserwacja ruchu naszych oczu może dostarczyć nam wiedzy o umyśle osoby badanej. Możemy na podstawie takich obserwacji wykrywać symptomy niektórych chorób, sprawdzać zmęczenie, a także naturalne skłonności i preferencje osoby badanej. Aby możliwe było wykorzystanie potencjalnych możliwości tkwiących w ruchach oczu musimy dysponować odpowiednimi środkami technicznymi umożliwiającymi monitorowanie pozycji gałek ocznych. Wydaje się na pierwszy „rzut oka", że metoda pomiarowa nie powinna nastręczać zbyt wielu trudności. W praktyce pomiar



Rysunek 2.3: Przykład ruchu oka śledzącego sinusoidalnie poruszające się pobudzenie

okazuje się być wyjątkowo trudny, głównie ze względu na charakter ruchów, które chcemy zmierzyć.

Pierwszym problem wynika z szybkości obserwowanych przez nas zjawisk. Typowa sakkada trwa nie dłużej niż kilkadziesiąt milisekund, a więc rozdzielczość czasowa obserwacji powinna być o rząd wielkości większa. To oznacza, że w przypadku pomiarów dyskretnych musimy zapewnić próbkowanie sygnału pozycji oka rzędu 1000 razy na sekundę.

Kolejną kwestią jest rozdzielczość przestrzenna urządzenia i jego zakres pomiarowy. Jeżeli chcemy obserwować mikrosakkady i mikrodryfty powinniśmy dysponować urządzeniem o rozdzielczości rzędu 1 minuty kątowej. Naturalne możliwości ludzkiego oka dają zakres ruchów $\pm 45^\circ$ w poziomie oraz $\pm 25^\circ$ w pionie.

Nie do pominięcia jest także sama metoda pomiaru, a dokładniej to jak oddziałuje ona na zmysły osoby badanej. Metody, które ingerują w żywą tkankę, a tym samym najczęściej wymagają, by osoba badana miała podane znieczulenie miejscowe (tzw. metody inwazyjne), nie nadają się do masowego stosowania, jednak to właśnie one wydają się być najdokładniejsze. Należałoby również zwracać uwagę na to, czy sama metoda nie powoduje zakłócenia w zachowaniu się badanego, na przykład oświetlanie oczu badanego światłem widzialnym nie wydaje się być dobrym rozwią-

zaniem. W trakcie kilkudziesięcioletniej historii badań nad ruchem ludzkiego oka wymyślono wiele różnych metod pomiarowych, bardziej i mniej skomplikowanych

2.3.1 Historia pomiarów

Spojrzenie na historię pomiarów ruchu ludzkiego oka [10] może być interesujące z dwóch przyczyn. Po pierwsze jest ciekawą opowieścią o postępie nauki, od prymitywnych metod w rodzaju przyczepionego do oka sznurka, po te najbardziej zaawansowane technologicznie wykorzystujące pola magnetyczne, czy też światło podczerwone. Po drugie (co z punktu widzenia niniejszej pracy jest ważniejsze) ta klasyfikacja pozwala na odniesienie wykorzystanej przez nas technologii pomiaru do innych technik. Jej samej poświęcony jest rozdział 2.3.2.

- **Bezpośrednia obserwacja.**
  Całkowicie bezużyteczna jeżeli chcemy obserwować całą złożoność zjawisk, które wiążą się ruchem oka. Ludzka obserwacja jest niedokładna przestrzennie i czasowo. Nieuzbrojone oko ludzkie nie jest w stanie dostrzec wszystkich ruchów, ze względu na ich niewielka amplitudę. Ponadto ewentualne zmęczenie eksperymentatora obserwacją może wpłynąć na wynik badań.

- **Mechaniczny zapis ruchów oczu.**
  Metoda polegająca na umocowaniu nici lub drutu do płytki metalowej bądź gipsowej. Tą natomiast przyklejano do oka osoby badanej (pod miejscowym znieczuleniem). Poprzez system bloczków nitka połączona była z narzędziem kreślącym na zwyczajnej kartce papieru. Dzięki temu kiedy pacjent ruszał gałkami ocznymi, na przesuwającej się kartce papieru można było rejestrować te ruchy w postaci wykresu pozycji oka. Druga metodą mechanicznego zapisu ruchu gałki ocznej jest wykorzystanie wypukłości rogówki, do której przyklejony był mały balonik. Ruchy oka powodowały zmianę ciśnienia w baloniku, co było graficznie rejestrowane.

- **Rejestracja odbitego światła na błonie fotograficznej.**
  Dzięki refleksyjnym właściwościom rogówki oraz samego dna oka możliwe jest rejestrowanie odbitego światłą. Metoda raczej mało dokładna i droga ze względu na duże koszta materiałów światłoczułych.

- **Filmowanie.**
  Metoda oparta na bezpośredniej rejestracji ruchu gałki ocznej. W epoce, w której nie była rozwinięta technika cyfrowa oznaczało to niezwykle żmudną pracę osoby, która na podstawie nagranego materiału filmowego musiała odtworzyć

położenie oka. Praca ta wydaje się być mało dokładna. Dzisiaj można by ją powierzyć komputerom, ale nadal pozostaje drugi problem związany z możliwością ruchu pacjenta lub tylko jego głowy. W takim przypadku to rozwiązanie staje się zupełnie bezużyteczna.

- **Metoda lusterkowo–fotograficzna.**
  Polega na przyczepieniu niewielkiego lusterka na powierzchnię rogówki oka. Zabieg i badanie wykonywane jest pod znieczuleniem. Rejestruje się światło odbite od lusterka na błonie fotograficznej. Metoda ta jest dosyć uciążliwa dla osoby badanej. Wymaga skomplikowanej procedury badawczej i samego do niej przygotowania. Niewątpliwą zaletą jest wysoka dokładność pomiarów ruchów oka (rzędu sekund kątowych).

- **Elektrookulografia.**
  Metoda polegająca na rejestracji różnicy potencjałów elektrycznych pomiędzy jedną i drugą stroną gałki ocznej. Podczas ruchu te dwa potencjały nieznacznie się zmieniają, dzięki czemu można określić położenie oka. Zaletą tej metody jest jej nieinwazyjność i możliwość przeprowadzania długich eksprymentów. Ponadto sama aparatura pomiarowa należy do najtańszych.

- **Metoda elektromagnetyczna**
  Stosowanie tej metody wymaga założenia specjalnych szkieł kontaktowych z wbudowqanymi cewkami indukcyjnymi. Osoba badana umieszczana jest w polu elektromagnetycznym (o kierunkach pinowych i poziomych). Pod jego wpływem w cewkach indukuje się sygnał elektryczny. Jego natężenie zależy od pozycji oka. Metoda ta charakteryzuje się dużą dokładnością (poniżej jednej minuty kątowej).

- **Metoda fotoelektryczna**
  Metoda ta wykorzystuje właściwość anatomiczną oka, polegającą na tym, że ruchy gałki ocznej powodują zmiany intensywności światła odbitego od rogówki. Współczesna wersja polega na tym, że oko oświetla się za pomocą kilku źródeł światła podczerwonego i rejestruje się jego natężenie po odbiciu od powierzchni oka. Natężenie światła zamieniane jest na napięcie elektryczne, a jego zmiany są odzwierciedleniem położenia oka zarówno w osi poziomej jak i pionowej. Metoda ta daje dokładność rzędu kilku minut kątowych, a samo oprzyrządowanie pomiarowe jest niewielkie i lekkie, co powoduje, że metoda ta nadaje sie do badań w warunkach naturalnych (początkowo była zresztą rozwinięta do badań procesów czytania u dzieci). Do zalet należy zaliczyć tak-

że jej nieinwazyjność i wysoką rozdzielczość przestrzenną pomiaru ruchu oka. Odmiana tej metody jest wykorzystywana w urządzeniu „Jazz".

2.3.2 Pomiary urządzeniem „Jazz"

Urządzenie pomiarowe „Jazz" jest tak naprawdę multi–sensorem, który potrafi rejestrować zestaw różnych sygnałów biofizycznych, w tym również pozycję oka w poziomie i w pionie. Pomiar ruchu oka wykonywany jest metodą fotoelektryczną. Wynikowy sygnał jest w rzeczywistości średnią sygnałów dla prawego i lewego oka. Taki sposób pomiaru jest użyteczny tak długo, jak ruchy oczu, które badamy są wykonywane równocześnie w tym samym kierunku. Taką naturę właśnie mają opisywane wcześniej interesujące nas ruchy.

Rysunek 2.4: Multisensor „Jazz"

Jedną z ważniejszych zalet tego urządzenia jest jego rozmiar i nieinwazyjny charakter jego stosowania. Zakłada się go na nos, podobnie jak okulary, przez co badany nie odczuwa żadnego dyskomfortu. Sensor jest bardzo lekki (zaledwie 35g) Ponadto w trakcie badania można nosić okulary, dzięki czemu osoby z wadami wzroku mogą być również badane z wykorzystaniem tego urządzenia.

„Jazz", ze względu na sposób jego działania (mierzy względną pozycję oka), nie wymaga właściwie żadnej sprzętowej kalibracji czy strojenia przed badaniem. Wystarczy go założyć i uruchomić test. Kalibracja wykonywana jest przez oprogramowanie na uzyskanym z „Jazz'a" sygnale pozycji oka. To sprawia, że urządzenie można wykorzystać niemal w każdych warunkach eksperymentalnych.

Urządzenie w obecnej wersji przesyła dane pomiarowe poprzez interfejs IrDA do stacji bazowej, której z kolei zadaniem jest przesłanie tych danych przez interfejs

szeregowy COM do komputera klasy PC. Układ pomiarowy jest zatem izolowany poprzez łącze podczerwone od komputera. Możliwe jest też przesyłanie danych droga radiową, a w najnowszej wersji systemu wykorzystuje się łącze światłowodowe.

Waga	35 g
Zasialanie	baterie (2×AA)
Częstotliwość próbkowania	1000 Hz
Zakres pomiaru w poziomie	± 45°
Zakres pomiaru w pionie	± 25°
Rozdzielczość przestrzenna	około 1'
Interfejs z komputerem	szeregowy

Tabela 2.1: Podstawowe parametry urządzenia „Jazz"

Urządzenie „Jazz" jest efektem 20–letniej pracy prof. Jana Obera z Instytutu Biocybernetyki z Polskiej Akademii Nauk. Jego badania dotyczyły psychofizjologii ruchu ludzkiego oka, a także zastosowań w praktycznych aplikacjach.

2.4 Praktyczne wykorzystanie ruchu oka

Ruchy naszych oczu są obecne przy każdej wykonywanej przez nas czynności. Jeżeli wykazują one wysoki stopień organizacji i zależności od procesów myślowych można sobie wyobrazić dziesiątki praktycznych zastosowań systemów monitorujących i analizujących ruch oka. Chcąc zaprowadzić wśród nich porządek można wprowadzić prostą klasyfikację.

Przede wszystkim systemy takie można podzielić na te, które badają ludzką psychikę i zajmują się samym człowiekiem (ocena kondycji umysłowej człowieka, diagnozowanie chorób, poznanie preferencji człowieka, itd.). Druga grupa zastosowań wiąże sie z wykorzystaniem samego monitoringu ruchu oka do ogólnie rozumianej optymalizacji pracy i/lub miejsca pracy człowieka. Zaliczyć tutaj można również systemy sterowania realizujące zadania komunikacji „człowiek–komputer".

2.4.1 Diagnozowanie chorób

Dzięki rozwijającym się badaniom nad samą naturą ruchu oczu potrafimy określić które reakcje na warunki eksperymentalne są typowe dla osób zdrowych, a które kwalifikują się jako patologiczne. Co więcej, niektóre z tych ostatnich skojarzono z występowaniem znanych dysfunkcji procesów poznawczych i narządu wzroku, dzięki czemu otrzymujemy skuteczne i w wielu przypadkach obiektywne narzędzie diagnostyczne.

Ta ostatnia cecha tego typu systemów wynika głównie z tego, że odpowiednie eksperymenty polegają na odruchach „niskiego poziomu", dzięki czemu trudnym zadaniem staje się sztuka oszukania badacza i tym samym wpłynięcie na wynik samego badania. Tak jak to było powiedziane wcześniej, reakcja okoruchowa na nowo pojawiające się bodźce jest w dużej mierze mimowolna. Człowiek może jednak wysiłkiem swojej woli starać się powstrzymać lub zmienić naturalny odruch, choć nie zawsze (tzn. nie w 100% przypadków) jest to skuteczne. Zwiększona obiektywność badań diagnostycznych może stać się zastosowaniem samym w sobie, wszędzie tam, gdzie nie można polegać na odpowiedziach badanego. Przykładem mogą być badania małych dzieci, a także badania przedpoborowe. Wśród wielu zastosowań z tej grupy należyw wyróżnić następujące przykłady.

1. **Badanie ostrości wzroku.**
   Badanie opiera się na wywołaniu reakcji oczopląsowej pojawiającej się w odpowiedzi na poruszającą się scenę. W trakcie ekspozycji różnych poruszających się scen, wyświetla się także elementy, które mają „przykuć" wzrok i zatrzymać reakcję oczopląsową, tak zwane „optotypy". W kolejnych scenach wyświetlane optotypy mają coraz to większe rozmiary. W pewnym momencie są tak duże, że osoba badana może je z łatwością zauważyć, co powoduje natychmiastowe przerwanie reakcji oczopląsowej i ruch sakkadyczny w kierunku jednego z optotypów. Na podstawie wielkości tegoż można określić ostrość wzroku osoby badanej. Istnieją zresztą inne warianty tego eksperymentu, wszystkie jednak bazują na pojawieniu się bądź też zniknięciu reakcji oczopląsowej.

2. **Parkinsonizm, choroba Alzcheimera, schizofrenia.**

   (a) Parkinsonizm — widoczne zwiększone prawdopodobieństwo występowania sakkad wielostopniowych w reakcji refiksacyjnej [5, 3].

   (b) Choroba Alzcheimera — zmniejszona umiejętność (zdolność) zmiany odruchowej reakcji refiksacyjnej. W eksperymencie badanemu każe się spojrzeć w przeciwnym kierunku do wystąpienia bodźca (tzw. anty–sakkada). U osób zdrowych około $70 \div 80\%$ prób kończy się sukcesem. U osób chorych obserwuje się wyraźne zachwianie tych proporcji.

   (c) Schizofrenia — eksperyment potrafi wykryć pewne symptomy występujące u osób chorych oraz tych, które są genetycznie spokrewnieni z osobami mającymi predyspozycje do zachorowania. Dzięki temu możliwe jest proste ocenienie samych skłonności (genetycznych) do rozwinięcia się pełno–objawowej choroby w pewnych sprzyjających warunkach. Więcej na temat można przeczytać w rozdziale 6.

3. **Problemy z czytaniem i rozumieniem tekstu**
   Dzięki badaniom nad procesami czytania i związanymi z nimi ruchami oczu możliwe jest wczesne wykrycie trudności u dzieci. To z kolei pozwala na zastosowanie kolejnych specjalistycznych badań i podjęcia odpowiedniej terapii.

2.4.2 Systemy monitorujące stan człowieka

Na szczególne zainteresowanie zasługuje cała klasa systemów, które ułatwiają automatyczne rozpoznanie stanu, w którym znajduje się człowiek. Są to przede wszystkim systemy czuwania, które potrafią na podstawie analizy ruchu oka (głównie częstości występowania sakkad) ocenić zmęczenie osoby badanej. Jest to wyjątkowo ważne wszędzie tam, gdzie powodzenie przedsięwzięcia zależy od umiejętności i przytomności umysłu jednego człowieka. Najczęściej przytaczanym przykładem jest monitorowanie stanu pilotów i kierowców. To od nich często zależy życie nie tylko ich samych, ale także pasażerów. Często jednak zdarza się, że prosta niedyspozycja jaką jest zmęczenie czy senność prowadzą w konsekwencji do katastrofy. Wykrycie na czas pojawiającej się senności pozwala na automatyczne wygenerowanie ostrzeżenia lub podjęcie innych niezbędnych działań.

W ramach programu VINTECH II realizowanego w ramach projektu europejskiego rozważa się także wykorzystanie monitorowania ruchu oka pasażerów linii lotniczych w celu wykrycia osób potencjalnie zagrażających bezpieczeństwu lotu. Mówiąc prostszym językiem: szuka się terrorystów, którzy zdradzaliby swoje podenerwowanie przed wykonaniem „zadania". Od kilkunastu ostatnich miesięcy tego typu zastosowania zdają sie mieć wielu zwolenników, a więc należy przypuszczać, że nie będzie brakowało funduszy na ich badania, co w konsekwencji powinno zaowocować pojawieniem się na rynku gotowych rozwiązań.

Zupełnie innym wykorzystaniem monitorowania ruchu oka są systemy ułatwiające ocenę postępów w nauce czynności, w których wzrok pełni ważną (chociaż nie koniecznie kluczową) funkcję. Przede wszystkim trzeba ponownie tutaj przywołać czytanie, a konkretnie jego naukę. System mógłby automatycznie informować o postępach i alarmować o sytuacjach, w których następuje nienormalne zmniejszenie szybkości uczenia się. Podobnie w trakcie nauki gry na instrumencie moglibyśmy ocenić postępy w nauce i zdecydować, kiedy osoba ucząca się może już wykonać dany utwór bez patrzenia na nuty.

Ponadto możliwe są dowolne inne zastosowania, które wykorzystują bezpośrednią obserwację i analizę ruchu oka lub też w trakcie wykonywania zaplanowanego eksperymentu. Na ich podstawie możliwa jest ocena natężenie różnych emocji i preferencji osoby badanej (testy polegające na krótkiej jednoczesnej projekcji dwóch

bodźców o różnej symbolice czy podtekstach).

2.4.3 Interfejs „człowiek–komputer”

Systemy realizujące zadanie komunikacji między człowiekiem a komputerem przychodzą najczęściej na myśl, kiedy myślimy o monitorowaniu ruchu oka. Wydaje nam się jednocześnie, że zadanie nie jest wcale trudne, co więcej wiele osób myśli o możliwości sterowania swoim własnym komputerem biurkowym, albo dowolnym innym systemem automatycznym przy pomocy wzroku.

Pomysł niestety nie daje się w prosty sposób zrealizować do wszystkich celów. Przyczyny mają charakter zarówno naturalnych ludzkich ograniczeń jak również technicznych problemów związanych z wykorzystywaniem konkretnej techniki pomiarowej.

Ograniczenia pierwszego rodzaju związane są z tak zwanym „skrzyżowaniem funkcji wzrokowych”: obserwowaniem i wskazywaniem (sterowaniem, np. jak myszą). Ten sam narząd wzroku musiałby służyć do dwóch diametralnie różnych celów jednocześnie: do obserwowania i przyswajania sobie informacji odczytywanych z urządzenia projektującego oraz do wskazywania wzrokiem miejsc i akcji, które należy wykonać. Wykonywanie obydwu czynności w tym samym czasie powoduje szybką dezorientację, dużą liczbę pomyłek a także po krótkim czasie zmęczenie.

Drugi z problemów ma charakter techniczny. Urządzenia, które mogłyby sie nadawać jako elementy takiego interfejsu z użytkownikiem powinny być ergonomiczne w użyciu. Takim systemem może być „Jazz”. To co jest siłą tego systemu, to znaczy fakt, że przyrząd mierzy względną pozycję oka, staje się w tym wypadku jego wadą. Nie jest możliwe bowiem dokładne określenie (bezwzględne) pozycji wzroku, ponieważ głowa jak i cała postać jest w ciągłym ruchu. To z kolei powoduje dekalibrację układu sensor–projektor i w konsekwencji jego całkowitą bezużyteczność nawet w izolowanej i przeznaczonej do tego przestrzeni.

Wprawdzie wykonanie klasycznego systemu interfejsowego wydaje się być całkiem trudnym przedsięwzięciem, to jednak monitorowanie ruchu oka można wykorzystać w nieco inny sposób. W pewnych zastosowaniach możliwe jest optymalizowanie miejsca pracy poprzez lepsze rozmieszczenie źródeł informacji wzrokowej (na przykład zegarów pokładowych w kabinie pilota). Owo polepszenie można by uzyskać poprzez uważną analizę położenia wzroku i najczęściej odwiedzanych miejsc. Ponadto system taki mógłby podpowiadać o szybkości uczenia się posługiwania wykorzystywaną aparaturą na podstawie czasu, które osoba ucząca się poświęciła każdemu z elementowi urządzenia.

Rozdział 3

Funkcjonalność systemu „UniJazz”

„UniJazz” jest programem umożliwiającym stworzenie i wykonywanie eksperymentów wizualnych z wykorzystaniem urządzenia monitorującego ruch oka o nazwie „Jazz”. Program został napisany w środowisku graficznym pod system operacyjny Windows 9x/2000, z wykorzystaniem biblioteki bezpośredniego dostępu do grafiki. Oprogramowanie w pełni korzysta z graficznych możliwości systemu operacyjnego, co oznacza, że jest łatwe i intuicyjne w użytkowaniu. Program jest kierowany do wszystkich osób, które są zainteresowane badaniem i wykorzystywaniem ruchów oka.

Do najważniejszych funkcji prograu „UniJazz” należy:

- projektowanie eksperymentów okoruchowych przy użyciu wewnętrznego dedykowanego języka; więcej o sposobie budowania i filozofii samego eksperymentu znajduje się w rozdziale 3.1, a na temat konstrukcji języka w rozdziale 4,
- wykonywanie wcześniej napisanych eksperymentów wraz z uruchomieniem i rejestracją danych z urządzenia „Jazz”, rozdział 3.2,
- przetwarzanie (kalibracja, filtrowanie), wizualizowanie, przechowywanie sygnałów, rozdziały 3.3 do 3.7.

Ze względu na szybkość działania przyrządu „Jazz” minimalne wymagania dotyczącego sprzętu komputerowego to komputer PC z procesorem klasy Intel Pentium III, z zegarem taktującym przynajmniej 700 MHz. System powinien posiadać przynajmniej 128 MB pamięci RAM. Zaleca się jednak stosowanie wydajniejszego zestawu komputerowego. W trakcie działania programu nie powinny działać inne aplikacje w tle, gdyż może to spowolnić i zakłócić pracę modułu odczytującego dane z przyrządu pomiarowego.

Urządzenie „Jazz” podłączanie jest do komputera za pomocą interfejsu szeregowego. Oprogramowanie umożliwia wybranie numeru portu, na którym będzie nawiązane połączenie. Więcej informacji na temat konfigurowania znajduje się rozdziale 3.9.

3.1 Budowa eksperymentu

Przebieg eksperymentu jest definiowany przez użytkownika programu za pomocą wbudowanego języka. Pozwala on na przygotowanie pojedynczego testu jako sekwencji następujących po sobie scen, przy ich czym kolejność może być modyfikowana w trakcie wykonywania eksperymentu, na przykład poprzez losowy ich wybór. Każda scena jest zbiorem pobudzeń prezentowanych na ekranie mających ustalone własności, np. kolor, wielkość, sposób poruszania się. Każda scena wyświetlana jest poprzez wykonanie specjalnej instrukcji `Show()` i pozostaje na ekranie (ciągle na nowo odrysowywana) na określony wcześniej okres czasu.

Każdy z obiektów składających się na definicję eksperymentu posiada zestaw nazwanych własności, które determinują jego zachowanie w trakcie eksperymentu (podczas wyświetlania na ekranie). Każdą z własności użytkownik może modyfikować w części definicyjnej, jak również w kodzie samego eksperymentu, co oznacza, że zachowanie się obiektów można zmieniać w trakcie trwania eksperymentu.

Definicje eksperymentów przechowywane są w plikach tekstowych (tekst niesformatowany) z rozszerzeniem „e”, np. „acuity.e”. W jednym pliku można przechowywać dowolną liczbę eksperymentów i innego typu definicji. Jedyną regułą, której należy przestrzegać jest różne nazewnictwo każdego z obiektów (*UWAGA: rozróżniane są małe i wielkie litery.*) Możliwe jest wywoływanie eksperymentów wcześniej napisanych, a dzięki instrukcji `import` możliwy jest podział dużego projektu na wiele plików i zawieranie innych definicji na poziomie plików. Przykładowa definicja eksperymentu znajduje się na listingu 3.1. Składa się ona z dwóch części: deklaracyjnej oraz definicyjnej. W pierwszej po słowie kluczowym `experiment` następuje nazwa eksperymentu (dokładniej nazwa obiektu) oraz następujące dalej deklaracje scen, które będą używane, a których definicje znajdują się gdzieś wcześniej w kodzie eksperymentu (przykład definicji sceny znajduje się na listingu 3.2). W części definicyjnej mamy kod procedury o nazwie `Run()`. Jest to procedura specjalna, której ciało powinno znajdować się w każdym eksperymencie, który będzie wykonywany bezpośrednio ze środowiska „UniJazz”. W momencie uruchomienia eksperymentu oprogramowanie poszukuje procedury o takiej właśnie nazwie i jej przekazuje sterowanie. Więcej na temat języka budowy eksperymentów znajduje się w rozdziale

```
experiment VCalibration {
           scene InitialScene as VInitialScene;
           scene top          as VTopScene;
           scene bottom       as VBottomScene;

           EyeY = 'True';

           proc Run()
           {
                InitialScene.TimeOut = 3000;
                InitialScene.Show();

                repeat 5 {
                         top.Show();
                         bottom.Show();
                }

                return;
           }
}
```

Listing 3.1: Przykład definicji eksperymentu

4.
Oprogramowanie „UniJazz” umożliwia otwieranie plików z eksperymentami, wyswietlenie ich na ekranie oraz umożliwia modyfikacje za pomocą prostego edytora. Możliwe jest też wykorzystywanie edytora zewnętrznego, w takim wypadku po każdorazowym uaktualnieniu pliku przed wykonaniem eksperymentu powinno się wywołać funkcję *File →Reload File*, które spowoduje ponowne załadowanie otwartego pliku. Za pomocą polecenia *File → New Experiment File* można utworzyć nowy plik do edycji eksperymentu.

3.2 Wykonywanie eksperymentów

Aby wykonać napisany wcześniej eksperyment należy wczytać go za pomocą polecenia *File → Open...* do programu. Po pojawieniu się okienka tekstowego możemy od razu uruchomić polecenie *Experiment → Run*. Spowoduje ono najpierw uruchomienie kompilatora. Jeżeli kompilacja zakończy się wykryciem błędu składniowego lub jakiegokolwiek innego wykonanie eksperymentu zostanie wstrzymane a na ekranie pojawi się okienko z wykazem błędów, które wykryto w trakcie kompilacji. W przeciwnym wypadku, to znaczy gdy tekst zostanie poprawnie skompilowany następuje uruchomienie eksperymentu zgodnie z aktualnymi ustawieniami dotyczącymi ekranu (zobacz rozdział 3.8. Jeżeli w zasięgu aktualnego pliku (to znaczy w plikach dołączonych za pomocą instrukcji `import`, albo w samym tekście tego pliku) znajduje

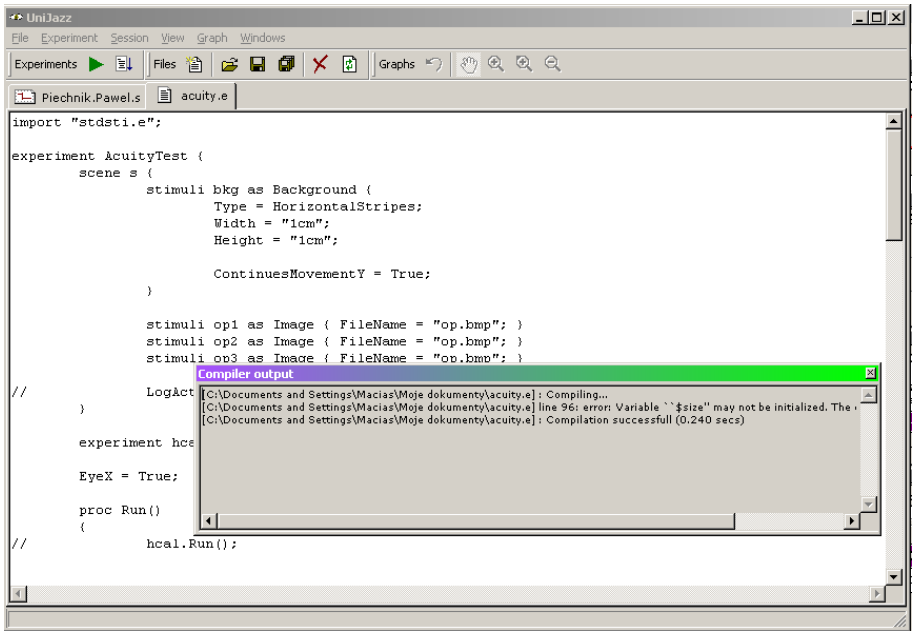
```
scene VTopScene {
      stimuli bkg as BlueBackground;

      stimuli cross as SmallCross {
              Top = "-10cm";
      }

      stimuli horline as Cross {
              Width = '20cm';
              Height = 0;
              GapWidth = '10cm';
              LineThickness = 1;
      }

      LogActivation = 'True';
      LogLabel = "Top";
      TimeOut = 1000;
}
```

Listing 3.2: Przykład definicji sceny

więcej niż jeden eksperyment użytkownik będzie miał możliwość wybrania jednego z nich przed uruchomieniem.

Możliwe jest też sprawdzenie poprawności składniowej kodu eksperymentu za pomocą osobnego polecenia *Experiment* → *Compile*, a także sprawdzenie wyników ostatniej kompilacji po otwarciu okienka *Experiment* → *Show Compiler Output*. Przed samym uruchomieniem eksperymentu sprawdzana jest dostępność urządzenia „Jazz”. Jeżeli program przy aktualnych ustawieniach nie wykryje podłączenia przyrządu do komputera wtedy użytkownik zostanie o tym poinformowany, a jednocześnie będzie mieć możliwość zatrzymania wykonania eksperymentu i co za tym idzie sprawdzenia sprzętu. Jeżeli natomiast wszystko działa prawidłowo następuje uruchomienie modułu zajmującego się zbieraniem danych z urządzenia, a zaraz potem następuje inicjalizacja podsystemu zarządzającego ekranem i wyświetlaniem grafiki podczas wykonywania eksperymentu. W kolejnym kroku program poszukuje w wybranym wcześniej eksperymencie procedury o nazwie `Run` i uruchamia moduł odpowiedzialny za wykonywanie skompilowanego kodu tej procedury (tzw. maszynę wirtualną).

W trakcie działania eksperymentu użytkownik może go w każdym momencie przerwać poprzez naciśnięcie klawisza *Escape*. Dotychczasowe zapisy sygnału nie przepadają w takim wypadku, można je oglądać w taki sam sposób jak po prawidłowo zakończonym eksperymencie. Po zakończeniu eksperymentu w głównym oknie programu pojawia się przebieg sygnałów monitorowanych podczas eksperymentu w postaci wykresu.



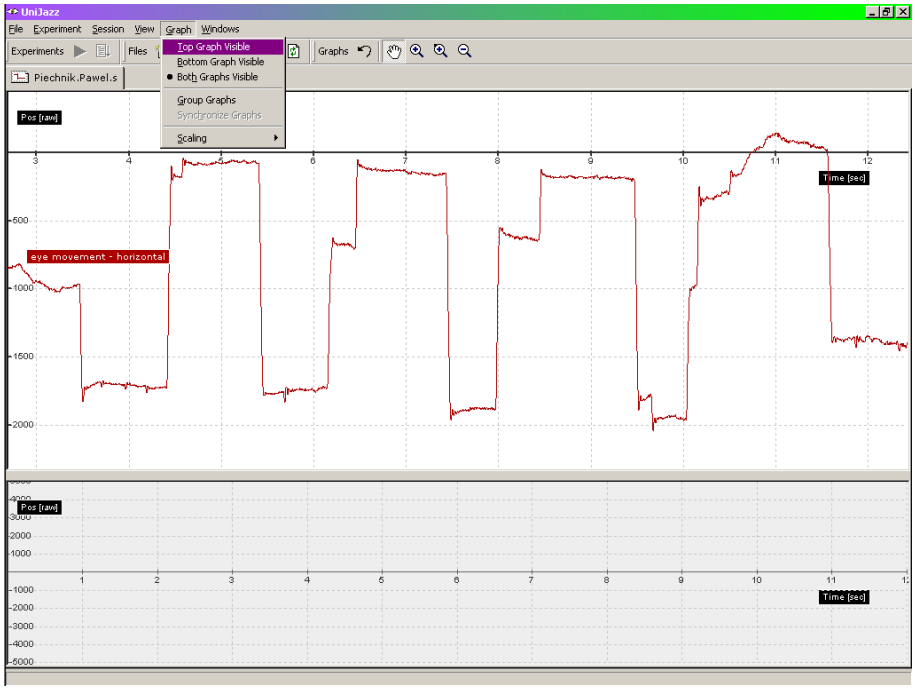
Rysunek 3.1: Wygląd edytora eksperymentu wraz z okienkiem wyników kompilacji

3.3 Przechowywanie i zarządzanie sygnałami

Jedną z ważniejszych umiejętności podczas obsługiwania programu jest sprawne manipulowanie wynikowymi syganałami, które pokazywane są zaraz po zakończeniu eksperymentu. W tym celu program wyposażono w specjalne kontrolki wykresów ułatwiające to zadanie. Standardowo okno z sygnałami składa się z dwóch części, każda z nich jest osobnym wykresem. Widoczność obydwu można regulować na dwa sposoby:

- poprzez zmianę pozycji paska rozdzielającego wykresy, dzięki czemu można dostosować wysokość podokienek do aktualnych potrzeb,
- poprzez ukrycie jednego z dwóch wykresów, w tym celu trzeba wykorzystać jedno z poleceń w menu *Graph*:
  - *Top Graph Visible* — widoczny tylko górny wykres,
  - *Bottom Graph Visible* — widoczny tylko dolny wykres,
  - *Both Graphs Visible* — widoczny obydwa wykresy.

Każdy z wykresów może być wyskalowany osobno, albo obydwa mogą pracować w trybie, w którym skalowanie jednego powoduje analogiczne przeskalowanie drugiego z nich. Aby uaktywnić tą funkcję należy zaznaczyć w menu *Graph* pozycję *Group Graphs*. Pozycjonowanie i skalowanie wykresu zależy od stanu, w którym się



Rysunek 3.2: Wygląd programu z otwartym przebiegiem ruchu oka, widoczne dwa wykresy i linia ich podziału

on znajduje. Możliwe są cztery stany, wszystkie dostępne w menu *Graph* → *Scaling* oraz w pasku narzędziowym o nazwie *Graphs*:

- *No Scaling* (symbol dłoni na pasku narzędziowym) — oznacza tryb pozycjonowania wykresu, możliwe są wtedy następujące akcje:
  - zmiana pozycji wykresu w oknie — następuje poprzez wciśnięcie i przytrzymanie lewego klawisza myszy oraz jednoczesne poruszanie wskaźnikiem, wykres będzie przerysowywany na bieżąco,
  - skalowanie osi wykresu — kiedy wskaźnik myszy znajdzie się nad jedną z osi wykresu (sygnalizowane przez zmianę wyglądu wskaźnika myszy na odpowiednio pionowe lub poziome strzałki) możliwe jest wtedy skalowanie jednej z osi, czynność tą wykonuje się poprzez wciśnięcie i przytrzymanie lewego klawisza myszy oraz jednoczesne poruszenie wskaźnikiem, odpowiednio w pionie lub poziomie.
- *Scaling Area Selector* (symbol lupki z plusem i szpikulcem na pasku narzędziowym) — w tym trybie możliwe jest powiększenie wybranego obszaru wykresu. Wybór obszaru następuje poprzez przyciśnięcie i przytrzymanie lewego klawisza myszy a następnie przesunięcie myszy w taki sposób, by rysowany na ekranie prostokąt pokrył się z obszarem do powiększenia. Zwolnienie przycisku

powoduje takie porzeskalowanie wykresy, aby wybrany wcześniej prostokątny fragment wypełnił całe okienko wykresu.

- *Zoomer In* (lupka z plusem na pasku narzędziowym) — pozwala na proste powiększenie obszaru, w środku którego nastąpi kliknięcie myszą
- *Zoomer Out* (lupka z minusem na pasku narzędziowym) — pozwala na pomniejszenie widocznego w oknie obszaru, to jest pokazanie obszaru większego. Akcja następuje poprzez kliknięcie myszą w dowolnym miejscu wykresu.

Rysunek 3.3: Wygląd paska narzędziowego dla wykresów

Możliwe jest także automatyczne przeskalowanie wykresu w taki sposób, ażeby zmieścić na ekranie cały przebieg zarówno na osi czasu jak również na osi wartości. Funkcja ta jest dostępna w menu *Graph* → *Scaling* → *Auto Scale*. Po wykonanym skalowaniu albo pozycjonowaniu wykresu użytkownik może w łatwy sposób powrócić do poprzedniego wyglądu wykresu poprzez wielopoziomowy mechanizm „*undo*”.

Wszystkie wymienione wcześniej operacje na wykresach są możliwe do wykonania jedynie na aktywnym wykresie. Aktywacja następuje poprzez kliknięcie myszą w dowolnym punkcie obszaru samego wykresu, a każdy aktywny wykres zmienia kolor swojego tła z szarego na biały. Jednocześnie może być aktywny co najwyżej jeden wykres.

Sygnały ruchu oka oraz wszystkie inne sygnały utworzone przez użytkownika mogą zostać zapisane do pliku wraz z informacjami o przebiegu samego eksperymentu, całość nosi nazwę *sesji*. Pliki z zapisem sesji posiadają rozszerzenie „s”, na przykład „hcal.s”.

3.4 Kalibracja sygnału

Kalibracja sygnału pozycji oka jest mechanizmem, dzięki któremu względna pozycja oka przeliczana jest na pozycję bezwzględną na podstawie warunków samego eksperymentu. Aby proces kalibracji uprościć bezpośrednio przed wykonaniem eksperymentu zaleca się uruchomienie eksperymentu kalibracyjnego dołączonego do oprogramowania w pliku *stdsti.e*. Znajdują się w nim dwie wersje eksperymentu: dla ruchu oka w pionie i w poziomie.Eksperymenty te nazywają się odpowiednio:

```
(...)
import ''stdsti.e'';
(...)
experiment MyExperiment {
           scene InitialScene as InitialScene;
           scene left         as LeftScene;
           scene right        as RightScene;

           experiment hcal as HCalibration;
           EyeX = True;
           (...)
           proc Run()
           {
                hcal.Run();
                InitialScene.TimeOut = 3000;
                InitialScene.Show();
                (...)
                return;
           }
}
(...)
```

Listing 3.3: Sposób wykorzystania eksperymentu kalibracyjnego

VCalibration oraz *HCalibration*. Listing 3.3 przedstawia przykładowy sposób wykorzystania takiego eksperymentu w kodzie użytkownika.
Po wykonaniu eksperymentu na ekranie pojawia się „surowy” przebieg sygnału pozycji oka odczytany z urządzenia „Jazz”. Kalibracja wykonywana jest dla każdego sygnału osobno i polega na przeliczeniu wartości sygnału według funkcji liniowej, która jest wyznaczana poprzez podanie wartości surowego sygnału i rzeczywistego położenia dla dwóch różnych punktów. Oprogramowanie przeprowadza automatycznie dobór tych punktów, dlatego właśnie ważne jest by stosować eksperyment kalibracyjny dołączony do programu. Dane do wyznaczenia funkcji kalibracyjnej są wynikiem uśrednienia kilku par (standardowo 10) rzeczywistych punktów znajdujących się na wykresie. Pary te są dobierane automatycznie przez program, jednak możliwa jest ich modyfikacja.

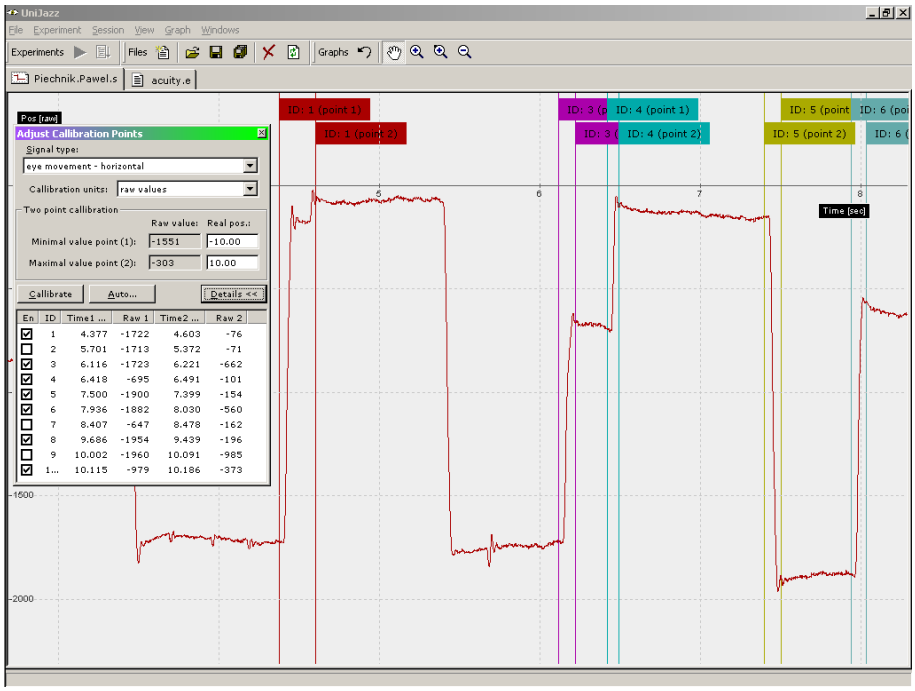
Sterowanie kalibracją sygnału odbywa się poprzez okienko narzędziowe dostępne w menu *Session* → *Units & Calibration*. Umożliwia ono następujące operacje:

- wybór sygnału do kalibracji (*Signal name*),
- wybór jednostek (*Display units*, które będą wyświetlane na osi wartości wykresów ze skalibrowanym sygnałem,
- wpisanie wartości rzeczywistej pozycji dla pierwszego (o minimalnej wartości surowej) i drugiego (o maksymalnej wartości surowej) punktu kalibracyjnego,

wartość powinna odpowiadać stosowanym jednostkom,

- ponowny automatyczny wybór punktów kalibracyjnych, z możliwością zmiany ich liczbę, oraz przesunięcia czasowego od początku sygnału

Zmiana jednostek na wykresie z uwzględnieniem punktów kalibracyjnych następuje po wciśnięciu przycisku *Change Units*. Jeżeli wykres na którym znajduje się sygnał będzie miał zmienione jednostki, wtedy użytkownik ma możliwość automatycznego przeskalowania wykresu w osi wartości. Po pokazaniu się okienka kalibracyjnego (o



Rysunek 3.4: Wygląd programu w trakcie kalibracji sygnału

nazwie *Units & Adjust Calibration Points*) na wykresie aktualnie wybranego sygnału pojawia się zestaw kolorowych znaczników. Początkowo każdy ze znaczników pokazuje pozycję automatycznie wybranych punktów kalibracyjnych. Punkty te są sparowane w taki sposób, że jeden z punktów w każdej parze odpowiada minimalnej wartości, a drugi z punktów wartości maksymalnej. Użytkownik może modyfikować położenie tych punktów, poprzez przesuwanie znaczników myszą. Po wciśnięciu przycisku *Details* możliwe jest obserwowanie szczegółów dotyczących każdej z pary, a także aktywowanie i dezaktywowanie każdej z nich. Para aktywna ma po lewej stronie na liście czarny symbol zaznaczenia (ang. *checkmark*). Dezaktywacja konkretnej pary powoduje, że oba punkty nie będą brane pod uwagę przy wyliczaniu średniej wartości, czyli wartości punktów kalibracyjnych. Spowoduje to także zniknięcie odpowiadającym im markerów na wykresie. W dowolnym momencie możliwa jest ponowna aktywacja punktów.

3.5 Wyznaczanie profilu prędkości

Jednym z ważnych parametrów sygnału pozycji oka jest profil prędkości czy przyspieszenia ruchu oka. Program „UniJazz” ma dedykowaną funkcję do wykonywania tego zadania w menu *Session → Create Velocity Profile*. Po wybraniu tej pozycji otwiera się okienko, w którym mamy następujące opcje: wybór sygnału, na podstawie którego wyznaczany będzie profil, wybór metody obliczania wynikowego sygnału, a także trzy dodatkowe przełączniki:

- **Automatically adjust graph scaling** — powoduje, że po obliczeniu profilu prędkości nastąpi automatyczne przeskalowanie osi wykresu, w którym nowy sygnał będzie wyświetlony. Przeskalowanie będzie wykonane w taki sposób, by cały sygnał zmieścił się w oknie.
- **Place in bottom graph** — powoduje umieszczenie wynikowego sygnału na dolnym wykresie zamiast w górnym jak to jest wykonywane domyślnie.
- **Absolute calculations** — spowoduje, że na sygnale wyjściowym będzie wykonana operacja wyznaczająca wartość bezwzględną. Jeżeli pole nie jest zaznaczone, prędkość obliczana jest ze znakiem.

Dostępne są dwie metody wyznaczania profilu prędkości. Pierwsza z nich (*Simple*) to wyznaczenie wartości ilorazu różnicowego dla każdej kolejnej pary punktów na wykresie.

$$v_i = \frac{f(t_i) - f(t_{i-1})}{t_i - t_{i-1}} \tag{3.1}$$

Druga metoda (*two point central differential*) jest połączeniem wyznaczania ilorazu różnicowego z filtrowaniem uśredniającym. Do obliczenia wartości prędkości w danym punkcie brane są pod uwagę dwa punkty leżące po obu stronach rozpatrywanego punktu w równych i ustalonych odległościach. Tę ostatnią użytkownik może modyfikować w polu nazwanym *Filter size*. Jeżeli wielkość tą oznaczymy jako s, to wartość każdego z punktów będzie wyznaczana zgodnie ze wzorem 3.2.

$$v_i = \frac{f(t_{i-n}) - f(t_{i+n})}{t_{i-n} - t_{i+n}} \tag{3.2}$$

$$n = \lfloor \frac{s}{2} \rfloor$$

3.6 Filtrowanie sygnału

Użytkownik może wybrać jedną z kilku metod filtracji sygnału po wykonaniu eksperymentu. Filtry dostępne są w menu *Session → Aplly Filter To Signal*. Zestaw wbudowanych w program filtrów ([1]) obejmuje następujące pozycje:

- filtr medianowy (*Median*) — poruszająca się maska o zadanej wielkości, wartość aktualnego punktu obliczana jest jako mediana punktów znajdujących się pod maską,
- filtr uśredniający arytmetyczny (*Avarage*) — poruszająca się maska o zadanej wielkości, wartość aktualnego punktu obliczana jest jako średnia arytmetyczna z wartości punktów znajdujących się pod maską,
- filtr dolnoprzepustowy (*Low Pass*) — o regulowanej wartości częstotliwości granicznej.

3.7 Zdarzenia

Zdarzenia to mechanizm umożliwiający na wprowadzanie znaczników czasowych do przebiegu pozycji oka. Znaczniki takie mogą byś generowane automatycznie w trakcie trwania eksperymentu jak również ich pojawienie się może być kontrolowane przez użytkownika. Do zdarzeń automatycznie wstawianych do sygnału należą:

- sakkady — okresy czasu, kiedy wykonywana jest sakkada,
- fiksacje — okresy czasu mijające od końca ostatniego zdarzenia typu „sakkada” do początku następpnego zdarzenia tego typu,
- kompleksy sakkadyczne — okresy czasu, w trakcie których następuje tzw. kompleks sakkadyczny, czyli sakkada wraz z towarzyszącymi jej sakkadami dodatkowymi (w tym sakkada korekcyjna, wielostopniowa sakkada), kompleksy wykrywane są na podstawie zdarzeń typu „sakkada” oraz czasów, które upływają pomiędzy nimi.

Zdarzenia mogą być również generowane z poziomu kodu eksperymentu albo za pomocą klawiatury w trakcie trwania eksperymentu. Mają one wtedy charakter znaczników punktowych wraz z nazwą. Użytkownik może je wstawić do tzw. dziennika zdarzeń, czyli listy wszystkich zdarzeń prowadzonej w trakcie eksperymentu, na dwa sposoby:

1. poprzez instrukcję języka `log` — po tym słówku w kodzie powinna wystąpić wartość, która będzie użyta jako nazwa do identyfikacji tego zdarzenia,

2. poprzez uruchomienie mechanizmu generującego zdarzenia w momencie wyświetlenia sceny na ekranie — możliwe za pomocą dwóch właściwości obiektów typu `scene`:

   - `LogActivation` — wartość `True` oznacza włączony mechanizm generowania zdarzenia w momencie pokazania się na ekranie tej sceny, `False`— mechanizm wyłączony (domyślna wartość)
   - `LogLabel` — wartość jest wykorzystywana jako nazwa automatycznie wygenerowanego zdarzenia

3. poprzez naciśnięcie przycisku na klawiaturze; jeżeli scena która jest w tym smamym czasie wyświetlana ma ustawiony niezerowy czas wygaśnięcia `TimeOut` to naciśnięcie klawisza spowoduje jedynie dodanie wpisu do dziennika zdarzeń; jeżeli jednak czas ten będzie mieć wartość zero, to po naciśnięciu przycisku klawiatiury scena jest deaktywowana i następuje przejście do wykonania kolejnej instrukcji w skrypcie tego eksperymentu, klawisz `Escape` jest wyróżniony, jego naciśnięcie powoduje zatrzymanie wykonywania eksperymnentu.

3.8 Ustawienia ekranu

Program daje możliwość definicji ustawień dotyczących trybu graficznego, w którym wyświetlany jest eksperyment. Ponadto aby zaprojektowany przez użytkownika eksperyment był poprawnie wyświetlany niezbędne jest także podanie wymiarów ekranu i odległości oczu osoby badanej od powierzchni monitora komputerowego, na którym będzie wyświetlany eksperyment. Oba typy ustawień dostępne są w menu *File → Settings → Screen*.

Użytkownik może wybrać rozdzielczość ekranu, na którym będzie wyświetlany eksperyment. Wprawdzie im wyższa rozdzielczość tym większa zdolność do pokazywania mniejszych i dokładniej odwzorowanych szczegółów, jednak temu wzrostowi towarzyszy zawsze czas potrzebny na wyrysowanie całej sceny. W krytycznych wypadkach może dojść do niepożądanych efektów przeskakiwania w trakcie prezentacji ruchu pobudzeń albo nawet do zakłóceń w odbiorze sygnału z urządzenia „Jazz”. Zaleca sie stosowanie rozdzielczości nie większej niż 800×600 przy wykorzystaniu standardowej konfiguracji sprzętu opisanej w na początku rozdziału 3. Możliwa jest zmiana głębi kolorów w polu *Color depth*.

Język służący do budowy eksperymentów umożliwia stosowanie standardowych bezwzględnych jednostek długości, takich jak milimetry. Aby elementy zdefiniowane za pomocą tych jednostek były poprawnie wyświetlane konieczne jest ustawienie wielkości ekranu. Służa temu pola *Width* oraz *Height*, zarówno wysokość jak i szerokość podaje się w milimetrach. Można też podać przekątną ekranu (w calach) i wybierając opcję *Calculate size using screen diagonal* program obliczy wymiary na podstawie standardowych proporcji ekranu 3:4. Dla poprawnego wyświetlania elementów, których wielkości zostały podane w radianach albo stopniach kątowych należy podać także odległość osoby badanej (jej twarzy) od ekranu monitora, na którym wyświetlany będzie eksperyment. Przycisk *Defaults* przywraca domyślne ustawienia.

3.9 Ustawienia urządzenia „Jazz”

Program umożliwia ustawienie dwóch podstawowych parametrów transmisji szeregowej do komunikacji z urządzeniem „Jazz”. Odpowiednie okienko dostępne jest pod funkcją *File* → *Settings* → *Jazz Device*. Pierwszym parametrem jest numer portu, do którego podłączone jest urządzenie. Drugi parametr to prędkość transmisji, w obecnej wersji używana jest wartość 115200.

W tym samym oknie można też modyfikować wartość dodatkowego parametru technicznego, który definiuje czas oczekiwania na dane pomiędzy kolejnymi odczytami z urządzenia „Jazz”. Domyślną wartością jest 40 ms. Nieumiejętna zmiana tej wartości może spowodować nieprawidłowe działanie programu, w tym błędne odczyty z urządzenia. Jedynym przypadkiem, kiedy manipulowanie tym parametrem jest uzasadnione jest posiadanie bardzo wydajnego sprzętu komputerowego. Wzrost wartości tego parametru powoduje, że program rzadziej zajmuje się zadaniem odczytywania danych, natomiast częściej odświeża ekran. Wzrost tego parametru jest ograniczony przez standardowy bufor danych i nie powinien przekroczyć wartości 250 ms. Przycisk *Defaults* przywraca domyślne ustawienia.

Rozdział 4

Język budowy eksperymentów

Program „UniJazz” posiada wbudowany maszynę wirtualną i kompilator prostego języka programowania, dzięki któremu możliwe jest przygotowywanie własnych eksperymentów. Kod źródłowy jest kompilowany do postaci ciągu kodów, który następnie jest wykonywany przez moduł wykonawczy również wbudowany w oprogramowanie. Forma skompilowana jest przechowywana wyłącznie w pamięci komputera, nie można zapisać jej do pliku.

Przebieg eksperymentu to właściwie wykonanie instrukcji zamkniętych w procedurę zdefiniowaną w ramach obiektu typu `experiment`. Oprogramowanie uruchamia procedurę o nazwie `Run`, a więc każdy eksperyment powinien mieć zaimplementowaną taką procedurę.

Projektowanie eksperymentu składa się z dwóch części. Po pierwsze użytkownik musi zdefiniować zestaw scen (albo inaczej: ekranów) zawierających pobudzenia o odpowiednich własnościach (atrybutach). Drugim krokiem jest implementacja samej procedury przebiegu badania, w której użytkownik definiuje w jakiej kolejności przygotowane wcześniej sceny mają być wyświetlane. Można też wtedy zmieniać zachowanie się całych scen jak również poszczególnych pobudzeń poprze modyfikację własności. W tym rozdziale opisane są wszystkie elementy i struktury składające się na język budowy eksperymentów.

4.1 Nazwy

Nazwy są używane do identyfikacji obiektów definiowanych przez użytkownika: pobudzeń, scen, eksperymentów, procedur i zmiennych. Dzięki nim możliwy jest też dostęp do własności wszystkich obiektów. We wszystkich nazwach rozróżniane są wielkie i małe litery, co oznacz, że nazwy `Run` oraz `run` są traktowane jako dwa różne identyfikatory.

Przy dostępie do obiektów poprzez identyfikatory (nazwy) przestrzegane są zasady zasięgu nazw. Nazwa obowiązuje w obrębie i na poziomie pojedynczego obiektu, jednak możliwy jest dostęp do obiektów wewnątrz obiektu zadeklarowanego w danym obiekcie. Stosuje się wtedy nazwy kwalifikowane, w których nazwy do kolejnych obiektów oddziela się znakiem „.” (kropki). Wyjątkiem od zasady zasięgu są nazwy zdefiniowane poza jakimkolwiek obiektem. Są one dostępne w każdym z obiektów, lecz jedynie jako obiekty bazowe (za pomocą kontsrukcji `as`) służące do definicji nowego obiektu o tym samym typie.

Każda nazwa jest formalnie ciągiem znaków odpowiadającym następującemu wyrażeniu regularnemu:

```
[_a-zA-Z][_a-zA-Z0-9]*
```

w szczególności nazwy nie mogą zawierać znaków białej spacji (spacja, tabulacja, znak końca wiersza i powrotu karetki), natomiast pierwszy znak nie może być cyfrą. Przykłady poprawnie skonstruowanych nazw:

```
_nazwa1_2
maly_mis_przez_kladke_bal_sie_isc
ExperimentOfTheDay
hcal.scena1.pobudzenie2.Wlasnosc6
```

Ostatni z przykładów pokazuje nazwę kwalifikowaną.

4.2 Wartości i literały

Własności oraz zmienne używane w definicjach procedur służą do przechowywania i dostępu do wartości. We wbudowanym języku „UniJazz'a” nie ma deklarowania typu zmiennych na poziomie kodu. To oznacza, że wszystkie niezbędne konwersje dokonywane są automatycznie, chociaż przewidziano możliwość sterowania konwersją. Dostępne są trzy typy wartości o następujących nazwach:

- `string` — łańcuch znaków,
- `int` — liczba całkowita ze znakiem o rozmiarze 32 bitów,
- `float` — liczba zmiennoprzecinkowa o rozmiarze 64 bitów.

Wymuszenie konwersji w kodzie można wywołać wykorzystując instrukcję o następującej składni:

```
nazwa_typu( wyrażenie )
```

gdzie `nazwa_typu` jest jednym z wymienionych wcześniej typów. Wyrażenia są opisane w rozdziale 4.8.1. Typowanie zmiennych i własności następuje także poprzez instrukcję przypisania im pewnego literału, bedącego symbolem wartości:

- literały typu `string` — tworzone są poprzez objęcie dowolnego łancucha znaków znakami pojedynczego (`'`) albo podwójnego (`"`) cudzysłowia,
- literały typu `int` — to po prostu liczby całkowite ze znakiem,
- literały typu `float` — to liczby w których określono część ułamkową, oddzieloną znakiem kropki (`.`).

Przykłady literałów i ich typy oraz wartości pokazane są na listingu 4.1:

```
"To jest łancuch znaków"        /* literał typu string */
'To też jest łańcuch znaków'    /* literał typu string */
#00ff00AA                       /* literał typu int, zapisany szesnastkowo*/
-7736                           /* literał typu int */
17772                           /* literał typu int */
-14.5                           /* literał typu float */
string( 14 )                    /* wartość typu string: "14" */
int( 3.14 )                     /* wartość typu int: 3 */
float( '15.6' )                 /* wartość typu float: 15.6 */
int( "bla bla bla")             /* wartość typu int: 0 */
```

Listing 4.1: Przykład literałów i ich wartości oraz typów

4.3 Własności

Własności to pewne nazwane atrybuty, które są wbudowane w obiekty o następujących typach: `experiment`, `scene`, `stimuli`. Własności takie determinują zachowanie danego obiektu zgodnie ze specyfikacją podaną w następnych rozdziałach. Każda własność ma swoją nazwę i wartość. Wartość własności może być modyfikowana z części deklaracyjnej kodu jak również w kodzie jakiejkolwiek procedury za pomocą instrukcji przypisania.

Własności mogą w specyficzny sposób interpretować przypisane im wartości. Na przykład, własności oznaczające wymiary jakiegoś elementu graficznego na ekranie przyjmują również wartości typu `string` (łańcuch znaków) wraz ze znajdującym przyrostkiem oznaczającym jednostkę. Własności takie będziemy nazywać specjalnymi, a kompletna lista ich typów znajduje się w tabeli 4.1.

Nazwa typu	**Lista wartości**	**Komentarz**
`bool`	`'True', 'False'`	wartości logiczne interpretowane jako prawda lub fałsz
`dim`	przyrostki liczb: `'px', 'cm', 'mm', 'deg', 'rad'`	jednostki oznaczające wielkości elementów, znaczenie przyrostków znajduje się w tabeli 4.2
`bkg_type`	`'Solid', 'VerticalStripes', 'HorizontalStripes'`	używana do rozróżnienia typu tła
`valign`	`'Top', 'Middle', 'Bottom'`	wyrównanie elementów graficznych w pionie
`halign`	`'Left', 'Center', 'Right'`	wyrównanie elementów graficznych w poziomie

Tabela 4.1: Typy własności specjalnych i możliwe wartości

Przyrostek	**Znaczenie (jednostka miary)**
`'px'`	piksele na ekranie, jednostka domyślna
`'cm'`	centymetry
`'mm'`	milimetry
`'deg'`	stopnie kątowe
`'rad'`	radiany
`''` (brak)	brak przyrostka oznacza uzycie domyślnej jednostki: pikseli

Tabela 4.2: Znaczenie przyrostków we własnościach specjalnych typu `dim`

4.4 Komentarze

W kodzie eksperymentu mogą znajdować się komentarze, to znaczy teksty, które są ignorowane przez kompilator. W języku programu „UniJazz" tworzy się je na dwa sposoby. Pierwszy typ komentarza rozpoczyna się od podwójnego znaku *slash*: `//`. Wszystkie znaki po tych symbolach aż do końca bieżącego wiersza są uznawane za komentarz i pomijane w procesie parsowania pliku. Drugi typ komentarza rozpoczyna się od dwuznaku `/*`, a kończy również dwuznakiem `*/`. Niemożliwe jest zagłębianie w sobie ani zazębianie się komentarzy żadnego typu. Przykład komentarza:

```
experiment MyExperiment { // To jest komentarz do końca tej lini

/* A to innego typu komentarz /* kończy sie
w˜następnej lini */
```

4.5 Definiowanie pobudzeń (*stimuli*)

Pobudzenia to elementy składające się na scenę. Wszystkie pobudzenia dostępne w programie „UniJazz" mają charakter graficzny. Z punktu widzenia języka budowy eksperymentu są to obiekty typu `stimuli`. Definicja pobudzenia może wyglądać w jeden z następujących sposobów:

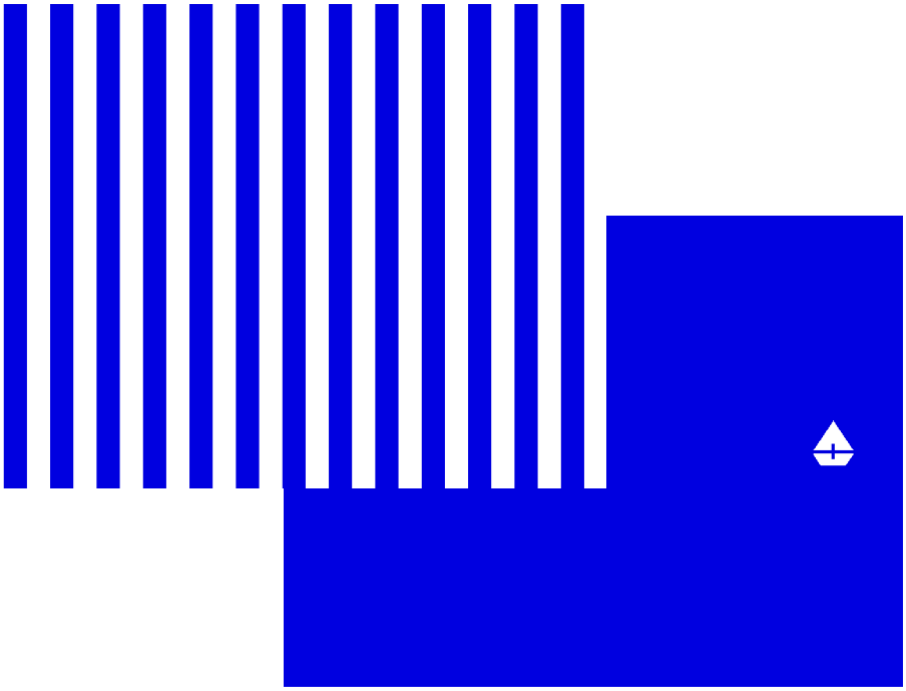
```
stimuli Nazwa as NazwaBazowa;
stimuli Nazwa as NazwaBazowa {
   // ustawienie własności obiektu pobudzenia
}
```

W obydwu wersjach `Nazwa` oznacza dowolny identyfikator obiektu, a `NazwBazowa` to identyfikator jednego ze standardowych pobudzeń, albo nazwa pobudzenia zdefiniowanego wcześniej. W tym drugim przypadku, obiekt pobudzenia tworzony jest z ustawieniami swoich własności takimi jak obiekt bazowy.

Program „UniJazz" oferuje kilka rodzajów pobudzeń. Każdy z nich jest krótko scharakteryzowany w kolejnych podrozdziałach:

- `Cross` — obiekt w kształcie krzyża,
- `Line` — linia na ekranie,
- `Circle` — okrąg o zadanym promieniu,
- `Image` — dowolny obraz wczytany z pliku graficznego,
- `Boat` — obiekt w kształcie łódki, do eksperymentów z pobudzeniami ruchomymi,
- `Text` — prosty obiekt wyświetlający tekst,
- `Background` — obiekt zajmujący się wypełnieniem całego ekranu, używany jako tło.

Kolejność definiowania pobudzeń w ramach sceny decyduje o kolejności rysowania na ekranie, stąd najczęściej na początku definiuje się obiekt typu `Background`. Definicje pobudzeń powinny znajdować się w obrębie definicji sceny, aby pobudzenia były wyświetlone na ekranie. Jednakże istnieje możliwość ich deklaracji poza jakimkolwiek obiektem, wtedy mogą one służyć jako obiekty bazowe dla innych pobudzeń. W pliku o nazwie *stdsti.e* dołączonym do oprogramowania znajdują się zdefiniowane częściej używane pobudzenia, np. krzyże o różnych rozmiarach, czy też niebieskie tło.



Rysunek 4.1: Przykładowe wyglądy ekranów podczas trwania eksperymentów

4.5.1 Własności wspólne dla wszystkich pobudzeń

Wszystkie obiekty pobudzeń mają zestaw wspólnych własności, które użytkownik może modyfikować. Ich lista wraz z komentarzem znajduje się w tabeli 4.3. Każdy z obiektów pobudzeń posiada także zestaw własności, które umożliwiają animację. Możliwe są do uzyskania trzy typy ruchów:

- **Ruch liniowy okresowy** (`RampMovement`)**.**
  Ruch o stałej zadanej prędkości i zadanej amplitudzie. Obiekt porusza się oscylacyjnie, punkty nawrotu wyznaczone są na podstawie własności określających amplitudę oraz początkowe położenie obiektu. Prędkość podawana jest w jednostkach miary na sekundę.

- **Ruch sinusoidalny okresowy** (`SinMovement`)**.**
  Ruch o prędkości zmieniającej się sinusoidalnie, o zadanej amplitudzie i okresie ruchu podawanym w sekundach.

- **Ruch liniowy ciągły** (`ContinuesMovement`)**.**
  Ruch o prędkości stałej, zadanej przez użytkownika.

Każdy typ ruchu może być aplikowany do obiektu niezależnie, dzięki czemu istnieje możliwość składania funkcji ruchu. Własności potrzebne do skonfigurowania animacji ruchów obiektów dostępne są w tabeli 4.4.

Nazwa	Typ	Znaczenie	Domyślnie
`Left`	`dim`	Określa położenie pobudzenia na ekranie w osi poziomej.Rzeczywista pozycja obliczana jest na podstawie wartości własności `HorizontalAlign`.	`0`
`Top`	`dim`	Określa położenie pobudzenia na ekranie w osi pionowej. Rzeczywista pozycja obliczana jest na podstawie wartości własności `VerticalAlign`.	`0`
`Width`	`dim`	Szerokość obiektu pobudzenia na ekranie.	`'1.2cm'`
`Height`	`dim`	Wysokość obiektu pobudzenia na ekranie.	`'2cm'`
`HorizontalAlign`	`valign`	Określa miejsce na ekranie w poziomie, od którego obliczna będzie ostateczna pozycja obiektu.	`'Center'`
`VerticalAlign`	`valign`	Określa miejsce na ekranie w pionie, od którego obliczna będzie ostateczna pozycja obiektu.	`'Middle'`
`Visible`	`bool`	Określa czy obiekt ma być wyrysowany razem ze sceną na ekranie.	`'True'`
`LineThickness`	`dim`	Określa grubość linii, którą będzie rysowany obiekt.	`'1mm'`
`LineSymetry`	`bool`	Określa czy grubość linii ma mierzona w pikselach ma być nieparzysta i rysowana tym samym symetrycznie. Jeżeli z obliczeń grubości liczba pikseli będzie nieparzysta wtedy program automatycznie doda jeden piksel.	`'True'`
`Color`	`int`	Definiuje kolor, w którym będzie obiekt rysowany na ekranie. Kolor jest podawany w postaci liczby całkowitej, w której trzy najmłodsze bajty oznaczają trzy składowe koloru: niebieski, zielony i czerwony. Najwygodniej zatem zapisywać kolor w postaci literału szesnastkowego o postaci `#RRGGBB`, gdzie `RR` to bajt oznaczający kolor czerwony, `GG` to kolor zielony, a `BB` to kolor niebieski.	`#FFFFFF`

Tabela 4.3: Podstawowe własności wspólne dla wszystkich pobudzeń

4.5.2 Pobudzenie *Cross*

Obiekt pobudzenia typu `Cross` jest wyświetlany w postaci krzyża równoramiennego. Stosuje się go jako podstawowe pobudzenie przyciągające wzrok i tym samym wywołujące reakcję refiksacyjną. Zwykle w miejscu przecięcia się ramion pozostaje nie zarysowana przestrzeń, która ma przyciągać, koncentrować wzrok — być punktem fiksacji. Poza standardowymi własnościami dostępne są także następujące:

- `GapWidth` oraz `GapHeight`. Obydwie są typu `dim` i określają odpowiednio szerokość i wysokość przerwy liczone od punktu środkowego krzyża, czyli miejsca, w którym przecinają się ramiona. Domyślne wartości to `'3mm'`.
- `IsBlinking`. Własność typu `bool` określająca czy krzyż ma naprzemiennie wyświetlać swoje ramiona. Domyślna wartość to `'False'`.
- `BlinkingFrequency`. Własność typu `float`. Określa częstotliwość z jaką krzyż wyświetla naprzemiennie swoje ramiona, jeżeli własność `IsBlinking` jest usta-

Nazwa	Typ	Znaczenie	Domyślnie
RampMovementX	bool	Ruch liniowy okresowy aktywny w osi poziomej	'False'
RampMovementY	bool	Ruch liniowy okresowy aktywny w osi pionowej	'False'
SinMovementX	bool	Ruch sinusoidalny aktywny w osi poziomej	'False'
SinMovementY	bool	Ruch sinusoidalny aktywny w osi pionowej	'False'
ContinuesMovementX	bool	Ruch liniowy ciągły aktywny w osi poziomej	'False'
ContinuesMovementY	bool	Ruch liniowy ciągły aktywny w osi pionowej	'False'
AmplitudeX	dim	Amplituda pozioma ruchu dla RampMovementX i SinMovementX	'20cm'
AmplitudeY	dim	Amplituda pionowa ruchu dla RampMovementY i SinMovementY	'20cm'
SpeedX	dim	Prędkość pozioma ruchu w jednostkach na sekundę dla ruchów linioowych	0
SpeedY	dim	Prędkość pionowa ruchu w jednostkach na sekundę dla ruchów linioowych	0
PeriodX	dim	Okres pełnego cyklu w ruchu sinusoidalnym w poziomie podany w sekundach	0
PeriodY	dim	Okres pełnego cyklu w ruchu sinusoidalnym w pionie podany w sekundach	0

Tabela 4.4: Własności umożliwiające ruch obiektów

wiona na 'True'. Częstotliwość jest podawana w cyklach na sekundę. Minimalna wartość to 1, maksymalna równa jest 50. Domyślna wartość to 2.

4.5.3 Pobudzenie *Circle*

Obiekt pobudzenia typu Circle jest pokazywany na ekranie jako okrąg o zadanym promieniu. Okrąg jest rysowany w wybranym kolorze, jednak dzięki metodzie tzw. *antyaliasingu* mogą się pojawić punkty w kolorach pośrednich między wybranym a aktualnym kolorem kolorem tła, na którym rysowane jest pobudzenie.

Jedyną dodatkową własnością w przypadku obiektów typu Circle jest promień okręgu. Można go modyfikować za pomocą własności o nazwie Radius. Jej typem jest dim, a domyślną wartością jest '1cm'.

4.5.4 Pobudzenie *Line*

Obiekty pobudzeń typu Line rysowane są jako linie o zadanych punktach początkowych oraz długościach w osiach X oraz Y. Posiadają dwie dodatkowe własności, które sterują umiejscowieniem punktu początkowego (Left oraz Top) względem samej linii. Oznacza to, że na przykład wartość Left może oznaczać punkt środkowy, albo skrajnie lewy lub też skrajnie prawy rysowanej linii. Przy ostatecznym umiejscowieniu linii są też brane po uwagę standardowe własności „wyrównania" obiektu. Dodatkowe własności noszą nazwy HorizontalAnchor oraz VerticalAnchor.

Ich typy to odpowiednio `halign` i `valign`, natomiast wartości domyślne to `Center` i `Middle`.

4.5.5 Pobudzenie *Image*

Pobudzenie typu `Image` pozwala użytkownikowi na wyświetlenie dowolnego obrazu graficznego w trakcie eksperymentu zapisanego wcześniej w pliku graficznym. Obsługiwane formaty plików graficznych to:

- **TGA** TrueVision Targa (musi posidać rozszerzenie `.tga`)
- **BMP** Windows Bitmap (`.bmp`)
- **PNM** Portable Anymap (`.pnm`)
  - `.pbm` — Portable BitMap (mono)
  - `.pgm` — Portable GreyMap (256 odcieni szarości)
  - `.ppm` — Portable PixMap (pełna 24 bitowa gama kolorów)
- **XPM** X11 Pixmap (`.xpm`)
- **XCF** GIMP native `(.xcf)`
- **PCX** ZSoft IBM PC Paintbrush (`.pcx`)
- **GIF** CompuServe Graphics Interchange Format (`.gif`)
- **JPG** Joint Photographic Experts Group JFIF (`.jpg` albo `.jpeg`)
- **TIF** Tagged Image File Format (`.tif` albo `.tiff`)
- **LBM** Interleaved Bitmap (`.lbm` albo `.iff`) Formaty: ILBM albo PBM (packed bitmap). HAM6, HAM8, and 24 bitowe typy nie są obsługiwane.
- **PNG** Portable Network Graphics (`.png`)

Obiekt typu `Image` posiada następujące dodatkowe własności:

- `FileName`
  Nazwa pliku z grafiką, własność typu `string`.
- `Smoth`
  Własność typu `bool` określająca czy obraz podczas skalowania ma zostać wygładzony. Domyślną wartością jest `'True'`.

- `HorizontalAnchor` oraz `VerticalAnchor`
  Własności odpowiednio typu `halign` i `valign`. Umożliwiają one wybranie punktu odniesienia w obrębie obrazka, który będzie umieszczony w miejscu oznaczonym przez standardowe własności `Left`, `Top`. Domyślnymi wartościami są odpowiednio `'Center'` oraz 'Middle'.

- `TransparentColor`
  Określa kolor, który nie powinien być kopiowany na ekran podczas rysowania obrazu. Jest to zatem kolor uznawany w obrazku za przezroczysty. Typem tej własności jest `int`, a domyślną wartością `0xFF00FF`.

Obrazy są przeskalowywane do wielkości określonych przez własności `Width` oraz `Height`. Jeżeli któraś z nich ma wartość `0`, wtedy wymiar, którego dotyczy jest rysowany w naturalnej wielkości, jaka posiada obraz w pliku. Dodatkowo, użytkownik może z jakiejś przyczyny zażądać, by przy następnym wyświetleniu sceny, w której znajduje się obraz został on ponownie załadowany z pliku i przeskalowany do odpowiednich rozmiarów. Za każdym razem w takim wypadku trzeba wykorzystać własność `ReloadImage` o typie `int` i przypisać jej wartość `1`.

4.5.6 Pobudzenie *Boat*

Pobudzenie typu `Boat` jest wykorzystywane w eksperymentach, w których samo pobudzenie wzrokowe porusza się. Wtedy znajomy kształt łódki powoduje silne zainteresowanie osoby badanej. Punktem fiksacji jest skrzyżowanie lini wewnątrz obiektu. Grubość tych linii można regulować poprzez własność `GapThickness`. Jest ona typu `dim`, a domyślna wartość to `'2mm'`.

4.5.7 Pobudzenie *Background*

Pobudzenie typu `Background` jest wykorzystywane jako tło dla eksperymentu. Ma ono szczególny charakter, ponieważ wypełnia cały ekran, a tym samym następujące własności nie maja znaczenia i są ignorowane: `Left, Top`. Tło może być wyświetlane na dwa sposoby. Ustawia się je przez własność specjalną typu `bkg_type` o nazwie `BackgroundType`.

1. `Solid` — jednolita powierzchnia o zadanym kolorze. W typ wypadku własności `Width, Height` są ignorowane.

2. `HorizontalStripes` albo `VerticalStripes` — naprzemienne pasy o różnych kolorach. Pasy mogą być odpowiednio poziome albo pionowe. Ich szerokość definiuje własność `Width`, natomiast wysokość (szerokość dla pasów poziomych)

`Height`. Pasy są wyświetlane naprzemiennie w dwóch kolorach, określonych własnościami `Color` oraz `StripeColor`, o domyślnych wartościach odpowiednio `0x0000E0` oraz `0xFFFFFF`.

Pobudzenie wyświetlające pasy jest używane do wywołania reakcji oczopląsowej u osoby badanej, poprzez wprawienie w ruch typu `ContinuesMovement`.

4.5.8 Pobudzenie *Text*

Pobudzenie typu `Text` pozwala w prosty sposób wyświetlić linię tekstu na ekranie za pomocą standardowej czcionki. Poprzez własność o nazwie `Label` możemy ustawić tekst do wyświetlenia. Własność ta jest typu `string`.

4.6 Definiowanie scen (*scene*)

Sceny są obiektami typu `Scene`, których zadaniem jest zebranie w jeden ekran wszystkich pobudzeń. Tutaj właśnie definiuje się wszystkie używane pobudzenia wzrokowe. Sceny pokazywane są na ekranie synchronicznie, co oznacza, że po ich pokazaniu następuje zawieszenie wykonywania kodu eksperymentu i systematyczne ciągłe odświeżanie ekranu, czyli odrysowywanie sceny na nowo.

Definicja sceny wygląda w jeden z następujących sposobów:

```
scene NazwaSceny {
      // Definicje pobudzeń wzrokowych
}
```

```
stimuli NazwaSceny as NazwaBazowa;
```

Drugą formę stosuje się, kiedy chcemy wykorzystać obiekt, który został zadeklarowany poza aktualnym zasięgiem nazw, na przykład w definicji eksperymentu korzystamy ze sceny definiowanej poza obiektem eksperymentu.

Aby wyświetlić scenę o nazwie `NazwaSceny` w eksperymencie projektowanym przez użytkownika trzeba zadeklarować lub zdefiniować taką scenę w obrębie obiektu eksperymentu a następnie wykonać instrukcję `Show` w kodzie procedury eksperymentu. Instrukcja ta zapisywana jest w taki sam sposób jak wywołanie bezparametrowej procedury obiektu sceny.
Obiekt sceny posiada następujące własności:

- `TimeOut`
  Własność typu `int` z domyślną wartością `0`. Określa czas, przez który scena

```
experiment MojEksperyment {
            scene MojaScena {
                  stimuli bkg as Background;
                  stimuli mycross as Cross { Left = "-10cm"; }
                  (...)
            }

            proc Run() {
                  (...)
                  MojaScena.TimeOut = random_int( 1000, 3000 );
                  MojaScena.Show();
                  (...)
            }
}
```

Listing 4.2: Przykład zdefiniwania i wyświtlenia sceny w eksperymencie

ma być wyświetlana na ekranie. Czas podawany jest w **milisekundach**. Jeżeli jej wartością jest `0` to wtedy scena jest pokazywana do momentu przerwania eksperymentu, czyli wciśnięcia przycisku *Escape*. W przeciwnym razie w momencie upłynięcia zadanego czasu następuje powrót do wykonywania kodu eksperymentu, w szczególności ekran nie jest w żaden sposób automatycznie czyszczony. Dzięki temu możliwe jest „płynne" przejście do następnej sceny.

- `LogActivation`
  Własność specjalna typu `bool`. Jej wartość domyślna to `'False'`. Określa, czy w momencie uaktywnienia sceny, to znaczy w momencie pokazania się na ekranie po raz pierwszy podczas wykonywania aktualnej instrukcji `Show` powinna ona zapisać do dziennika eksperymentu zdarzenie. Takie zdarzenie może być powiązane z pojawieniem się pobudzenia wzrokowego na ekranie.

- `LogLabel` Własność typu `string`. Jej wartość jest traktowana jako etykieta dla zdarzenia wygenerowanego podczas uaktywnienia sceny.

Istotną rolę przy definiowaniu obiektów pobudzeń w ramach jednej sceny pełni ich kolejność pojawiania się, bowiem w takiej samej kolejności będą one rysowane na ekranie. W szczególności ważne jest by obiekty tła (typu `Background`) były definiowane przed wszystkimi innymi pobudzeniami wzrokowymi. W przeciwnym wypadku nastąpi zakrycie niektórych obiektów i w konsekwencji nigdy nie pojawią się one na ekranie.

4.7 Definiowanie eksperymentów (*experiment*)

Obiekt type `experiment` ma za zadanie zebranie wszystkich niezbędnych elementów potrzebnych do wykonania eksperymentu: pobudzeń wzrokowych, scen wraz z pobudzeniami oraz procedur wraz z kodem samego eksperymentu. Definicja eksperymentu przybiera następującą postać:

```
experiment NazwaEksperymentu {
    // Definicje pobudzeń
    // Definicje scen
    // Deklaracje eksperymentów
    // Ustawienie własności eksperymentu
    // Definicje procedur
}
```

Aby możliwe stało się wykorzystanie innego, wcześniej napisanego eksperymentu konieczna jest jego deklaracja w ciele eksperymentu, który z niego korzysta. Wtedy deklaracja taka ma postać:

```
experiment Nazwa as NazwaEksperymentuZewnetrznego;
```

Eksperyment w ten sposób zadeklarowany możemy wykorzystać, na przykład uruchamiając go w procedurze napisanej wewnątrz tego eksperymentu.

W ramach eksperymentu możliwe jest zadeklarowanie wielu procedur, jednak jedna z nich jest wyróżniona i nosi nazwę `Run`. Tak jak to było już wyjaśniane to właśnie procedura o tej nazwie jest wykonywana w momencie uruchomienia eksperymentu. Jej brak spowoduje, że eksperyment natychmiast po uruchomieniu zakończy się. Jednakże w ramach tej procedury możliwe jest wywoływanie innych, należy jednak pamiętać, że wiąże się to zawsze z pewnym dodatkowym narzutem czasowym, co niekorzystnie odbija się na rygorach czasowych eksperymentu. Dlatego też zaleca się wpisywanie kodu eksperymentu w obrębie jednej procedury, najlepiej standardowej o nazwie `Run`.

Obiekt eksperymentu posiada tylko dwie własności, określające które z sygnałów ruchu oka mają być monitorowane w trakcie wykonywania eksperymentu. Są one nazwane `EyeX` oraz `EyeY`, a ich typ to `bool`. Domyślne wartości obydwu własności to `'False'`. Jeżeli w obrębie jednego eksperymentu zadeklarowany jest inny eksperyment, i jego kod zostaje wywoływany, to ignorowane są wszystkie własności eksperymentu wewnętrznego. Oznacza to, że nie jest możliwe uruchomienie dodatkowego kanału danych, albo jego usunięcie w trakcie wykonywania eksperymentu.

4.8 Definiowanie procedur (*proc*)

Procedury są elementem języka, który pozwala na zdefiniowanie przebiegu samego eksperymentu. Można je deklarować wyłącznie w obrębie samego obiektu eksperymentu (`experiment`). Wewnątrz procedury możliwe są proste konstrukcje sekwencyjnego programowania strukturalnego, w skład których wchodzą: wyrażenia przypisania, arytmetyczne, wywołanie funkcji i procedur, powtarzanie fragmentu kodu, wybór warunkowy, wyjście z procedury. Ponadto wewnątrz procedur można używać zmiennych. Deklaracja procedury wygląda następująco:

```
proc NazwaProcedury()
{
     // Ciało procedury
}
```

W aktualnej wersji kompilatora „UniJazz" nie możliwe jest przekazywanie zmiennych do procedur.

4.8.1 Wyrażenia

Wyrażeniem bedziemy nazywać jedną z następujących konstrukcji językowych:

- instrukcja przypisania,
- działanie arytmetyczne,
- instrukcja konwersji typów,
- literał,
- wywołanie fuknkcji.

Instrukcja przypisania służy do nadawania wartości nazwanej własności albo zmiennej. Przyjmuje ona postać:

```
<nazwa> = <wyrazenie>;
```

gdzie `<nazwa>` jest nazwą własności albo zmiennej, natomiast `<wyrazenie>` jest dowolnym wyrażeniem.

Możliwe są dwa podstawowe działania arytmetyczne: dodawanie i odejmowanie. Ze względów na ograniczenia samego parsera nienawiasowane wyrażenia arytmetyczne mogą ograniczać się wyłącznie do jednego działania.

```
<wyrazenie> + <wyrazenie>
<wyrazenie> - <wyrazenie>
- <wyrazenie>
```

Operacja konwersji typów ma następującą postać:

```
nazwa_typu( <wyrazenie> )
```

gdzie `nazwa_typu` jest identyfikatorem jednego z podstawowych typów: `string`, `int`, `float`.

Wywołanie funkcji lub procedury to podanie nazwy tejże i bezpośrednio za nią wyspecyfikowanie w nawiasach listy parametrów. W chwili obecnej jest dostępna jedna funkcja wbudowana o nazwie `random_int( min, max [, suffix] )`, która przyjmuje dwa parametry obowiązkowe i jeden opcjonalny. Funcja oblicza całkowitą liczbę pseudolosową mieszczącą się w przedziale $[min, max]$. Do powstałej liczby dodawany jest przyrostek `suffix`, będący parametrem opcjonalnym. Przykładem jego zastosowania może być zadanie wylosowania położenia obiektu na ekranie z zakresu -10cm do 2cm. Aby to wykonać wystarczy wywołać funkcję:

```
random_int( -10, 2, 'cm')
```

i przypisać jej wartość pewnej własności.

4.8.2 Instrukcja *repeat*

Instrukcja `repeat` pozwala na wielokrotne wykonanie fragmentu kodu. Składnia instrukcji jest następująca:

```
repeat <wyrazenie_ile> { <instrukcje> }
```

gdzie `<wyrazenie_ile>` jest wyrażeniem, którego wartość jest typu numerycznego. Instrukcje `<instrukcje>` zostaną powtórzone tylokrotnie ile wynosi wartość wyrażenie:

```
int( <wyrazenie_ile> )
```

4.8.3 Instrukcja *switch*

Instrukcja `switch` pozwala na wybór fragmentu kodu do wykonania w zleżności od wartości pewnego wyrażenia:

```
switch ( <wyrazenie_wartosc> ) {
case <const1>:
     <lista_instrucji1>
case <const2>:
     <lista_instrucji2>
(...)
[ default:
     <lista_instrucjiX> ]
}
```

Wartość wyrażenia obliczana jest przed wykonaniem instrukcji. Następnie porównywana jest ta wartość ze stałymi wyspecyfikowanymi jako `<constN>`. Jeżeli któraś z nich będzie równa wartości obliczonego wyrażenia, wtedy następuje wykonanie kodu znajdującego się po dwukropku. Jeżeli żadna stała na liście nie jest równa wartości, to wtedy jest wykonywana lista instrukcji `<lista_instrukcjiX>`. Jeżeli tej części nie ma, wykonywana jest następna instrukcja po instrukcji `switch`.

4.8.4 Instrukcja *return*

Instrukcja `return` kończy wykonywanie aktualnej procedury i powraca do kodu wywołującego tą procedurę. Jeżeli nie ma na stosie więcej procedur, wtedy kończony jest eksperyment. Każda procedura powinna posiadać instrukcję `return`, chociaż jej brak nie jest wychwytywany przez parser jako błąd składniowy.

```
return;
```

4.8.5 Instrukcja *log*

Instrukcja `log` pozwala na umieszczenie w dzienniku eksperymentu zdarzenia. Składnia instrukcji wygląda następująco:

```
log <wyrazenie>;
```

Wartść wyrażenia będzie uznana za etykietkę zapisywanego zdarzenia.

4.8.6 Zmienne

W obrębie procedur możliwe jest korzystanie ze zmiennych, czyli nazwanych miejsc w pamięci, do których możemy zapisywać, i z których możemy odczytywać wartości. Zmienne nie mają określonego typu ani miejsca deklaracji. Deklaracja następuje poprzez użycie zmiennej. Trzeba przy tym uważać, by nie odczytywać wartości ze

zmiennej wcześniej nie użytej, gdyż może to spowodować wystąpienie błędu (parser nie potrafi wychwycić wszystkich sytuacji z tym związanych). W przypadku zmiennych obowiązują ogólne zasady nazewnictwa, jednak każde odwołanie do zmiennej musi być poprzedzony znakiem `$`. Przykłady użycia zmiennych:

```
$repeat_count = 10;
$size = random_int( 10, 100, "mm" );
image1.Width = $size;
image1.Height = $size;
log 'Wielkość ' + $size;
$zmienna = 1 + random_int( 1, 10 );
```

4.9 Importowanie innych plików (*import*)

Możliwe jest wykorzystanie plików z kodem i definicjami wcześniej utworzonych przez użytkownika. W tym celu trzeba zaimportować do aktualnego pliku zawartość pliku zewnętrznego. Wykonuje się to poprzez instrukcję `import`, która jest dyrektywą dla kompilatora, aby skompilował on inny plik i przeniósł wszystkie wynikowe obiekty do tego pliku, w którym wywołanie owej instrukcji nastąpiło. Składnia wymaga jedynie podania ścieżki dostępu do pliku:

```
import literal_typu_string;
```

gdzie `literal_typu_strng` jest stałą łańcuchową, zawierającą ścieżkę do pliku, na przykład:

```
import 'stdsti.e';  // import standardowych definicji i~eksperymentów
import 'c:\scr\plik1.e';
```

Rozdział 5

Implementacja

5.1 Użyte narzędzia i biblioteki

Oprogramowanie „UniJazz" zostało zaimplementowane w języku programowania *C++* w środowisku programistycznym *Microsoft Visual C++* w wersji *6.0* [4, 9]. Już użyte narzędzia programistyczne wymuszają niejako podejście obiektowe w projektowaniu oraz implementacji. W taki też sposób zostało ono wykonane, co więcej w przypadku klas powiązanych z elementami języka skryptowego do budowy eksperymentó, filtrami i w wielu innych przypadkach wykorzystano autorski rozszerzony model obiektowy. Opiera się on kilku klasach bazowych, które deklarują operację tworzenia kopii obiektu, i implementują nazywanie klas, a także samych instancji obiektów. Ponadto udostępniają one mechanizm wymiany nazwanych własności. Wszystkie klasy zaprojektowane z użyciem tych udogodnień są zarejestrowane w globalnym obiekcie zwanym „fabryką klas". Dzięki takiemu podejściu oraz istnieniu osobnej „fabryki" dla modułu kompilatora samo zadanie kompilacji znacznie upraszcza się i automatyzuje.

Razem z kompilatorem *Visual C++* wykorzystywana jest także biblioteka *MFC*, ale wyłącznie w części zajmującej się obsługą graficznego interfejsu użytkownika. Wyjątkiem jest często stosowana klasa napisów `CString`. Wszystkie kontrolki GUI wykorzystane w oprogramowaniu „UniJazz" pochodzą z tejże biblioteki lub są z nich dziedziczone, w tym również dwie nowe napisane na podstawie bazowej klasy `CWnd` [4].

Oprogramowanie wykorzystuje bibliotekę *SDL*, która oferuje niskopoziomowy dostęp do wielu standardowych podsystemów komputerowych jak obraz, dźwięk, klawiatura, mysz i inne (oficjalna strona internetowa: `www.sdllib.org`). W obecnej wersji oprogramowania tylko dwa z nich są wykorzystywane: grafika i odczyty z klawiatury w trakcie wykonywania eksperymentu. Dzięki takiemu rozwiązaniu możliwe

Plik	Nazwa biblioteki	Opis
`sdl.dll`	SDL	jądro biblioteki SDL
`sdl_gfx.dll`	SDL_gfx	zestaw procedur rysujących
`sdl_image.dll`	SDL_image	wczytywanie formatów graficznych
`zlib.dll`	SDL_image	obsługa kompresji grafiki
`libpng1.dll`	SDL_image	obsługa formatu PNG
`jpeg.dll`	SDL_image	obsługa formatu JPEG

Tabela 5.1: Biblioteki niezbędne do uruchmienia programu „UniJazz”

jest przełączanie ekranu w tryb pełnoekranowy, a także wybór trybów wyświetlania grafiki (w tym rozdzielczości ekranu) oraz wykorzystanie sprzętu wspomagającego grafikę, a więc nowoczesnych i powszechnie stosowanych kart graficznych z wbudowanymi akceleratorami grafiki. Osobną cechą samej bibliotekami jest jej przenaszalność pomiędzy platformami systemowymi.

Wraz z biblioteką *SDL* oprogramowanie korzysta z usług dówch dodatkowych: `SDL_gfx` oraz `SDL_image`. Pierwsza z nich implementuje podstawowy zestaw procedur rysujących na ekranie, dzięki drugiej możliwe jest wczytywanie wielu różnych formatów plików graficznych.

5.2 Wielowątkowość i pobieranie danych

Aplikacja „UniJazz” wykorzystuje mechanizm wielowątkowości dostępny w systemie operacyjnym *Windows* ([4]). Bezpośrednio przed uruchomieniem samego eksperymentu tworzony jest odrębny wątek, którego zadaniem jest odczyt i przetwarzanie danych pochodzących z urządzenia „Jazz”. Kod tego modułu iteracyjnie powtarza następujące czynności:

1. Odczyt surowych danych z portu, do którego podłączone jest urządzenie.
2. Dekodowanie ramek w odczytanych danych, również rekonstrukcja sygnału.
3. Aplikowanie filtru dolnoprzepustowego oraz algorytmu poszukującego sakkad w sygnale.
4. Przepisanie danych do odpowiednich strumieni przechowujących pozycję o ruchu oka.
5. Uśpienie wątku na zadany czas (domyślnie 40 ms).

Klasa zajmująca się dekodowaniem ramek, jak również algorytm znajdywania sakkad pochodzi z laboratorium prof. J. Obera, zespołu, który stworzył urządzenie „Jazz”.

5.3 Parser, kompilator i maszyna wirtualna

Moduł zajmujący się kompilacją i wykonaniem eksperymentów składa się z kilku klas. Pierwsza z nich to analizator składniowy, który jest wykorzystywany przez parser. Analizator został utworzony w języpku *C++* i zoptymalizowany co do szybkości działania.

Drugim elementem jest parser połączony z kompilatorem. Parser, również napisany w *C++* działa „z góry na dół", co oznacza, że następuje analiza gramatyczna poprzez zagłębianie się w produkcje symboli. Cały proces rozpoczyna się od symbolu startowego gramatyki a kończy na symbolach terminalnych. Parsowanie dzięki temu zostało zaimplementowane w postaci wywołań funkcyjnych.

Kompilator jest sprzężony z parserem i generuje kodu eksperymentu oraz ustawienie wszystkich niezbędnych struktur danych do wykonania tego kodu. W momencie wykrycia błędu następuje awaryjne przerwanie procesu analizy i przekazywana jest jednocześnie informacja o błędzie: numer linii w analizowanym pliku oraz opis słowny błędu. Jeżeli kompilacja się powiedzie, wygenerowany kod może zostać wykonany przez interpreter kodów zamknięty w odrębnej klasie.

5.4 Kontrolki GUI

W ramach projektu „UniJazz" zaprojektowane zostały dwie dodatkowe kontrolki graficznego interfejsu użytkownika (GUI). Pierwsza z nich to kontrolka wykresu. Pozwala ona na swobodne manipulowanie przebiegami na wykresie, powiększanie i pomniejszanie wybranych jego obszarów. Ponadto nakładki na samą kontrolkę umożliwiły stworzenie prostego w użytkowaniu mechanizmu kontroli punktów kalibracyjnych.

Drugą z kontrolek jest tzw. *grid*, który pozwala na wyświetlanie i edycję informacji pogrupowanych na kolumny i wiersze. Kontrolka pozwala na zarządzaniem wielkością komórek, a także ich wyglądem i sposobem edycji (dostępne są w tej wersji trzy: zwykła edycja tekstowa, poprzez wybór z listy oraz wybór koloru).

```
PFile            :=    PStimuli PFile;
                 |     PScene PFile;
                 |     PExperiment PFile;
                 |     PImport PFile;
                 | ;
PProperty        :=    idstring "=" PPropertValue ";";
PPropertyValue   :=    float;
                 |     number;
                 |     idstring;
                 |     string;
PStimuli         :=    "stimuli" idstring "as" idstring PStimuliTail;
PStimuliTail     :=    ";";
                 |     "{" PStimuliBody "}";
PStimuliBody     :=    PProperty PStimuliBody;
                 |     ;
PScene           :=    "scene" idstring PSceneRest;
PSceneRest       :=    "as" idstring PSceneTail;
                 |     PSceneTail;
PSceneTail       :=    ";
"                |     "{" PSceneBody "}";
PSceneBody       :=    PProperty PSceneBody;
                 |     PStimuli PSceneBody;
                 |     ;
PExperiment      :=    "experiment" idstring PExperimentRest;
PExperimentRest  :=    "as" idstring PExperimentTail;
                 |     PExperimentTail;
PExperimentTail  :=    ";
"                |     "{" PExperimentBody "}" ;
PExperimentBody  :=    PProperty PExperimentBody;
                 |     PScene PExperimentBody;
                 |     PProc PExperimentBody;
                 |     PExperiment PExperimentBody;
                 |     ;
PImport          :=    "import" STRING ";";
PProc            :=    "proc" idstring "(" PParamList ")" "{" PProcBody "}";
PParamList       :=    ;
PProcBody        :=    PStatement PProcBody;
                 |     ;
PStatement       :=    PSimpleStatement ";";
                 |     PBlockStatement;
PSimpleStatement :=    PExpression;
                 |     PReturn;
                 |     PLog;
                 |     ;
PSetStatement    :=    LValue = PExpression;
PExpression      :=    "(" PExpression ")";
                 |     PProcCall;
                 |     LValue;
                 |     PConst;
                 |     PTypeConv;
                 |     PSetStatement;
                 |     PPlusStatement;
                 |     PMinusStatement;
```

Listing 5.1: Gramatyka języka budowy eksperymentów (część 1)

```
PProduct            :=    RValue "*" PExpression;
                    |     PExpression;
PProduct            :=    PExpression * PExpression;
                    |     PExpression / PExpression;
                    |     PExpression;
RValue              :=    PConst ;
                    |     PVariable ;
                    |     PProcCall ;
                    |     PPropValue;
PPlusStatement      :=    RValue "+" PProduct;
PMinusStatement     :=    RValue "-" PProduct;
PLog                :=    "log" PExpression;
PReturn             :=    "return" PExpression;
PProcCall           :=    QName "(" PArgList ")";
PArgList            :=    PExpression;
                    |     PExpression "," PArgList;
                    |     ;
QName               :=    _IDSTRING;
                    |     _IDSTRING "." QName;
                    |     ;
LValue              :=    PProperty2;
                    |     PVariable;
PProperty2          :=    QName;
PVariable           :=    QName _VARNAME;
                    |     _VARNAME;
PTypeConv           :=    _int "(" PExpression ")";
                    |     _float "(" PExpression ")";
                    |     _string "(" PExpression ")";
PSwitchStatement    :=    "switch" "(" RValue ")" "{" PSwitchBody "}" ";
" PSwitchBody       :=    PCaseStatement PSwitchBody ;
                    |     PDefaultStatement ;
                    |     ;
PCaseStatement      :=    "case" PConstValue ":" PProcBody;
PDefaultStatement   :=    "default" ":" PProcBody;
PRepeatStatement    :=    "repeat" RValue "{" PProcBody "}";
PNameStatement      :=    PProcCall ";";
                    |     PSetStatement;
Name                :=    idstring NameRest;
NameRest            :=    "." idstring NameRest;
                    |     ;
PProcCall           :=    "(" PArgList ")" ;
PArgList            :=    RValue PArgTail;
                    |     ;
PArgTail            :=    "," RValue PArgTail;
                    |     ;
PSetStatement       :=    "=" RValue ";";
PConstValue         :=    _STRING;
                    |     _FLOAT;
                    |     _NUMBER;
RValue              :=    PConstValue;
                    |     Name PProcCall;
                    |     Name PPropValue;
PNameRValue         :=    PProcCall;
                    |     ;
```

Listing 5.2: Gramatyka języka budowy eksperymentów (część 2)

Rozdział 6

Praktyczne zastosowania

6.1 Program edukacyjny

Oprogramowanie „UniJazz' powstałe w ramach niniejszej pracy jest przeznaczone dla wszystkich osób zainteresowanych zagadnieniami ruchu oka. Może ono spełniać zadania programu edukacyjnego dla tychże osób. Jest narzędziem, dzięki któremu możliwe jest poznanie wielu aspektów ruchu oka. Stało się to możliwe przede wszystkim dzięki łatwej obsłudze programu, a także szerokim możliwościom eksperymentowania z ruchem oka. Użytkownik tego zestawu (oprogramowanie oraz urządzenie „Jazz") może z powodzeniem budować nowe testy i modyfikować już istniejące.

Z myślą o edukacyjnym charakterze programu wraz z nim powstawało szereg „standardowych" eksperymentów. Wszystkie dołączone są do oprogramowania i pozwalają na obserwację podstawowych zjawisk związanych z ruchem oka takich jak:

- ruchy sakkadyczne — dokładność sakkad, czas reakcji, sakkady korekcyjne, antysakkady
- ruchy wolne typu *smooth pursuit* — inicjalizacja i finalizacja ruchów, reakcja na zanikający bodziec wzrokowy, reprogramowanie ruchu,
- ruchy wolne wywołane obserwacją poruszającej się sceny — inicjalizacja i finalizacja, reprogramowanie ruchu, test na ostrość widzenia.

Wszystkie eksperymenty zostały przekazane i zaprojektowane z dużą dbałością o szczegóły oraz podstawy teoretyczne przez prof. J. Obera.

6.1.1 Sakkady

Eksperymenty dotyczące sakkad zostały zebrane w pliku o nazwie „saccades.e". Wszystkie znajdujące się tam eksperymenty korzystają ponadto z eksperymentu

kalibracyjnego zdefiniowanego w pliku „stdsti.e". Ponadto dla uproszczenia opisu zjawisk, eksperymenty dotyczą wyłącznie dla ruchu oka w osi poziomej. We wspomnianym pliku znajdują się następujące eksperymenty:

- **Antysakkada**
  Eksperyment bada możliwości kontroli naturalnego odruchu refiksacyjnego. Początkowo wyświetlany jest jeden środkowy punkt fiksacji (krzyżyk). Po upływie losowego czasu punkt ten znika i jednocześnie pojawia się w innym miejscu: po lewej albo po prawej stronie, zawsze jednak w równej odległości od środka ekranu (i początkowego punktu fiksacji). Osoba badana ma za zadanie spojrzeć w dokładnie przeciwna stronę do tej, po której pokazał się punkt refiksacyjny. Po upływie około pół sekundy w miejscu, w którym osoba badana powinna spoglądać pojawia się punkt fiksacji. Powtarzając ten eksperyment wielokrotnie w trakcie jednej sesji można wyznaczyć odsetek udanych prób powstrzymania odruchu refiksacyjnego.

- **SimpleRefixation**
  Na ekranie wyświetlanych jest pięć punktów fiksacji: jeden po środku, oraz po dwa z każdej ze stron. Wszystkie oprócz skrajnych oddalone są od siebie w równych odległościach. Odległość skrajnych krzyży od środka pola widzenia nie przekracza 20 stopni, aby nie wywoływać ruchów głowy osoby badanej. Na początku punkt środkowy jest wyróżniony (nieco większy i grubszy od pozostałych). Po upływie losowego czasu następuje przerzucenie wyróżnionego punktu na jeden z sąsiednich w stosunku do środkowego. Powtarzając ten cykl możemy obserwować zdolność naszego układu wzrokowego do nauki pozycji, do której następuje refiksacja. Dzieje się tak dlatego, że punkty fiksacji pozostają ciągle w tych samych pozycjach, a zmienia się jedynie czas fiksacji początkowej, a więc moment zmiany punktu fiksacji.

- **AdvancedRefixation** Początkowo na ekranie pokazywany jest na środku pola widzenia jeden punkt fiksacji: krzyż. Po upływie pewnego zadanego czasu wybierana jest losowa pozycja po lewej albo po prawej stronie środka pola. Wygaszany jest środkowy punkt, a zapalany ten o wybranej pozycji. Osoba badana nie może nauczyć się stałej pozycji, w którym pojawia się punkt refiksacyjny. Uczy się natomiast momentu czasu, kiedy następuje zmiana pozycji punktu. Można zatem spodziewać się zmniejszonego czasu reakcji.

- **GapSaccade**
  Przebieg tego eksperymentu jest podobny do **AdvancedRefixation**, z tą różnicą, że pozycje nie są losowe, lecz ustalone, a za to moment refiksacji jest

losowy. Zasadnicza różnica polega jednak na tym, że pomiędzy wygaszeniem środkowego punktu a momentem pokazania drugiego z nich następuje krótka pauza, trwająca około 100 ÷ 200 ms. Ten krótki okres powoduje, że umysł osoby badanej ma czas na zwolnienie zasobu uwagi i jest gotowy do przydzielenia jej kolejnej czynności, w tym wypadku obserwacji punktu refiksacyjnego. W efekcie można obserwować skrócone czasy latencji sakkadycznej (okresu czasu upływającego od pojawienia się bodźca do rozpoczęcia ruchu sakkadycznego).

- **OverlapSaccade** Eksperyment symetryczny względem **GapSaccade**. Punkt refiksacyjny pojawia się kiedy punkt środkowy jest cały czas wyświetlany. Dopiero po krótkim okresie czasu jest on wygaszony. Można w tym wypadku obserwować zwiększony czas reakcji refiksacyjnej, gdyż w momencie pojawienia się drugiego bodźca nasza uwaga skupiona jest na tym pierwszym.

- **DiffAttractivity_Distance, DiffAttractivity_Size**
  Testy te badają atrakcyjność bodźców o różnych charakterystykach: odległości od początkowego punktu fiksacji oraz wielkości. Można przekonać się, że osoba badana wybiera najczęściej z dwóch prezentowanych obiektów, ten który znajduje się bliżej jej punktu fiksacji, albo jest bardziej wyraźny (np. większy).

W pliku „att_fix.e" znajdują się dwa testy umożliwiające obserwację zachowania się oka w momencie, kiedy usuniemy punkt fiksacji na dłuższy okres czasu. Widoczne są wtedy sakkady korekcyjne. Zjawisko to wydaje się być dosyć dziwnym i czeka jeszcze na naukowe wyjaśnienia. Pewne jest jednak, że musi istnieć poza wzrokowy układ stabilizujący wzrok na nieistniejącym, wyimaginowanym obiekcie.

6.1.2 Ruchy wolne nadążne

Ruchy wolne nadążne (ang. *smooth pursuit, tracking*) wywołane są przez poruszający się obiekt, na którym potrafimy skupić nasz wzrok. Wszystkie eksperymenty tego typu znajdują się w pliku „smooth.e". Zdefiniowane są tam następujące eksperymenty:

- **SmoothPursuit_InitializationAndFinalization**
  Początkowo nieruchomy obiekt na ekranie zaczyna się poruszać w jedną z losowo wybranych stron z losową stałą prędkością. Porusza się ruchem wahadłowym i po upływie losowego okresu czasu powraca skokowo do pozycji początkowej. W trakcie tego eksperymentu można obserwować anatomię ruchu wolnego: jego inicjalizację oraz zakończenie. W drugiej wersji tego badania,

nazwanego **SmoothPursuit_InitializationAndFinalization2** zakończenie ruchu polega na prostym zatrzymaniu się obiektu.

- **SmoothPursuit_DisappearingTarget**
  W trakcie badania na ekranie porusza się obiekt ruchem wahadłowym. W losowych momentach czasu znika on z ekranu i pojawia się po krótkim czasie ponownie, w tym miejscu, które wynika z wykonywanego ruchu. Dobierając różne czasy absencji obiektu (od 100 do 500 ms) można obserwować różne reakcje naszych oczu. Przy krótkich czasach nieobecność pobudzenia wydaje się być niezauważona.

- **SmoothPursuitInitialization_LeftDispl**
  SmoothPursuitInitialization_RightDispl
  W eksperymentach tych bada się wyłącznie moment rozpoczęcia ruchu oka w dwóch różnych warunkach. Początkowo na środku ekranu wyświetlany jest punkt, a następnie tuż przed rozpoczęciem ruchu w prawo przemieszczany jest on ruchem skokowym nieznacznie w lewo, albo w prawo w stosunku do położenia centralnego.

- **TwoObjects_EqVelocity**
  Dwa obiekty zaczynają poruszać się w przeciwnych kierunkach z jednakową prędkością stałą. Badanie pozwala obserwować reakcję oka na taką sytuację, najczęściej krótki oczopląs.

- **TwoObjects_DiffVelocity**
  Dwa obiekty zaczynają poruszać się w przeciwnych kierunkach z różnymi prędkościami. W eksperymencie tym można się przekonać, że znacznie silniejszym bodźcem okazuje się być szybki ruch obiektu.

6.1.3 Obserwacje poruszającej się sceny

W pliku „optokinesis.e" znajdują się trzy eksperymenty, które obrazują zjawiska związane z inicjalizacją, zakończeniem oraz reprogramowaniem ruchu optokinetycznego. Obserwować można reakcję oczopląsową oka na poruszające się dwubarwne paski. Paski te zaczynają się poruszać ze stanu spoczynku, zatrzymują się albo zmieniają kierunek swojego ruchu w zależności od zastosowanego testu. Przykład praktycznego zastosowania ruchu optokinetycznego znajduje się w pliku „acuity.e", gdzie odnaleźć można procedure testu na ostrość widzenia. Opisany jest on w rozdziale 2.4.1.

6.2 Zaburzenia ruchu oka u osób chorych na schizofrenię

6.2.1 Wprowadzenie

Schizofrenia to schorzenie neurologiczne, które dotyka najczęściej ludzi młodych. Skutki choroby zwykle powodują, że osoba jest odsunięta od społeczeństwa, czuje wyobcowanie, a przede wszystkim cierpi na zaburzenia procesów poznawczych. Przyczyny ujawnienia i rozwinięcia się choroby są kombinacją bardzo wielu czynników środowiskowych jak również predyspozycji genetycznej do danej osoby.

Udało się wykazać, że istnieje związek pomiędzy występowaniem odmian polimorficznych genu kodującego enzym o nazwie *catechol–O–methyltransferase* a sprawnością wykonywania testów neuropsychologicznych. Te z kolei są wykorzystywane przy ocenie stopnia zaawansowania choroby i używane jako wskaźniki jej występowania. U osób chorych obserwuje się jednocześnie zaburzenia w ruchu i stabilizacji oka. Widoczne są one głównie jako występowanie:

- dużej liczby sakkad goniących (*catch–up saccades*) w ruchu oka typu *smooth pursuit*,
- sakkad „wtrąconych" w ruch oka oraz w trakcie fiksacji oka na nieruchomym punkcie,
- krótkotrwałych (ponizej 150 ms) przeskoków wzroku (dwa ruchy sakkadyczne), tzw. *square wave jerks* (SWJ),
- opóźnień reakcji oka na zmianę charakterystyki ruchu punktu śledzonego.

Nasuwa się zatem wniosek, że istnieje pewien związek pomiędzy występowaniem choroby oraz zwiększoną liczbą nieprawidłowości w ruchu oka [7]. Tym samym istnieje przesłanka, aby wykorzystać badanie ruchu oka jako diagnozę choroby, a zaburzenie ruchu oka uczynić jednym z symptomów występowania schizofrenii. Ze względu na genetyczne podłoże choroby identyczne zmiany w genach mogą pojawić się również u osób spokrewnionych. Biorąc po uwagę fakt, iż wykazano istnienie związku pomiędzy występowaniem mutacji genetycznych w wymienionym wcześniej genie [8] a pojawianiem się zaburzeń w ruchu oka istnieje teoretyczna możliwość zbudowania taniego, prostego i szybkiego badania przesiewowego, które umożliwiałoby ocenę predyspozycji osoby badanej do zachorowania.

Niniejsze badania miały na celu sprawdzenie istnienia związku pomiędzy występowaniem anomalii w ruchu oka podczas zadania *smooth pursuit* a zachorowalnością

na schizofrenię. Jednocześnie udało się pokazać, iż stworzone oprogramowanie „UniJazz" pozwala na szybkie przygotowanie odpowiedniego testu wzrokowego, następnie jego wykonanie i opracowanie wyników.

6.2.2 Osoby badane

W badaniu wzięły udział trzy grupy osób. Do pierwszej z nich należą pacjenci Szpitala Wojskowego w Bydgoszczy oraz Kliniki Akademii Medycznej w Bydgoszczy ze zdiagnozowaną schizofrenią (4 osoby, 2 kobiety, 2 mężczyzn). Druga grupa to dwóch pacjentów z tych samych ośrodków, u których w trakcie badań podejrzewano wystąpienie choroby, jednak nie postawiono jeszcze ostatecznej diagnozy. Trzecią grupą są osoby zdrowe, 4 kobiety i 6 mężczyzn. Każda z osób w tej ostatniej grupie otrzymała osobny identyfikator skłądający się z litery oraz cyfry. Posiadanie tej samej litery oznaczać będzie pokrewieństwo pierwszego stopnia u tych osób.

6.2.3 Pomiar ruchu oka

Wszystkie osoby badane poddane zostały takiemu samemu eksperymentowi. Ich zadanie polegało na śledzeniu pobudzenia typu `Boat` poruszającego się poziomym ruchem oscylacyjnym, sinusoidalnie zmiennym. Odległość między oczami osoby badanej a monitorem, na którym wyświetlane były pobudzenia wynosiła około 60 cm. Eksperyment rozpoczynał się badaniem kalibracyjnym (`HCalibration`) a następnie wykonywany był eksperyment `SchizTest` znajdujący się w skrypcie o nazwie *schizofrenia.e*.

Właściwy eksperyment pomiarowy rozpoczyna się od nieruchomego pobudzenia umieszczonego na środku ekranu. Następnie po upływie 1,5 sekundy pobudzenie rozpoczyna swój ruch z okresem sinusoidy równej 2 sekundy. Po upływie losowego czasu, jednak nie większego niż 6 i nie mniejszego niż 3 sekundy, następuje zmiana okresu ruchu na 5 sekund. Cały cykl powtarzany był 10–krotnie. Pomiar oka ruchu oka wykonywany był przy pomocy urządzenia Jazz z częstotliwością 1000 Hz. Mierzono ruch oka wyłącznie w osi poziomej. Pomiary wykonano na kolorowym monitorze matrycowym TFT o przekątnej ekranu 14,1 cala.

6.2.4 Analiza i wielkości mierzone

Uzyskane przebiegi analizowano za pomocą oprogramowania „UniJazz". Dla każdej osoby badanej brano pod uwagę po dwa cykle ruchów (dla każdego z dwóch okresów sinusoidy) rozpoczynając od pierwszego cyklu zaraz po kalibracji. U dwóch osoób analizowano zamiennie dwa dalsze cykle, ze względu na duże zakłócenia w sygnale

spowodowane poruszeniem się osoby badanej w trakcie wkonywania eksperymentu. Nie wykorzystywano pozostałych cykli, ponieważ u większości badanych było widać zmęczenie monotonnością badania, co powodowało, że rozglądając się wprowadzali oni zaburzenia do ruchu oka.

Analizie poddano wyłącznie ruch oka. Zliczano wystąpienia sakkad w trakcie trwania analizowanych fragmentów ruchu (SAC), a także wystąpnienia zakłóceń typu *SWJ*. W tabeli 6.1 podano dla każdej osoby badanej liczbę wystąpień anomalii każdego z typów dla obydwu okresów sinusoidy (A — faza szybka, B — faza wolna). Dodatkowo zapisano czas trawania każdej fazy ruchu w sekundach (kolumny **Czas A** oraz **Czas B**). Tabela 6.2 zawiera natomiast dla każdej osoby zliczenia anomalii każdego z typów oddzielnie, ale zsumowane dla obydwu faz ruchów (szybkiej i wolnej). Ponadto umieszczono w niej średnią częstotliwość występowania każdej z anomalii w trakcie przeporwadzonego eksperymentu (kolumny **F SWJ** i **F SAC**). Częstotliwości podane są w postaci średniej liczby wystąpień danego zdarzenia na jednostkę czasu, sekundę. Ostatnia tabela (6.3) zawiera średnią częstotliwość występowania anomalii ruchu oka dla trzech grup osób oraz pacjentów z podejrzeniem schizofrenii (P1 i P2). Oprócz średnich częstotliwości podano także wartości odchyleń standardowych.

Osoba	**SWJ (A)**	**SWJ (B)**	**SAC (A)**	**SAC (B)**	**Czas A** [s]	**Czas B** [s]
D1	0	0	46	31	11,6	17,3
D2	1	3	29	46	8,5	17,8
D3	2	3	22	30	9,4	18,4
D4	0	0	32	37	10,0	16,7
P1	0	0	37	34	9,2	15,7
P2	0	0	19	11	5,5	6,9
A1	1	0	22	27	7,5	15,3
A2	0	1	24	24	8,7	18
B1	1	0	19	16	8,4	16,7
B2	0	0	16	27	10,4	15,9
C1	0	0	20	21	7,3	14,3
C2	1	1	34	32	8,5	16,7
E1	0	2	44	37	10,6	14
E2	0	2	33	26	11,0	16,3
E3	0	1	14	19	10,1	15,5
E4	0	1	24	14	8,7	13,4

Tabela 6.1: Liczba anomalii dla osób chorych zdiagnozowanych (D) i podejrzanych (P) oraz dla osób zdrowych (pozostali)

6.2.5 Wyniki

Najbardziej zaskakującą obserwacją, jest fakt, że jedna z osób (E1) zakwalifikowana do grupy zdrowych miała najbardziej „regularne anomalie" w trakcie trwania

ruchu *smooth pursuit*. Głównie były to sakkady goniące, a ich częstotliwość jest zdecydowanie najwyższa wśród wszystkich badanych. Pobierzna jakościowa obserwacja i analiza zachowania się oka w trakcie fiksacji (co nie jest przedmiotem tych badań) wykazała jednak, że w odróżnieniu od osób zdiagnozowanych E1 nie miała problemów z zadaniem fiksacji wzroku na nieruchomym pobudzeniu.

Drugą osobą z najwyższą częstotliwością wsytępowania sakkad jest jedna z podejrzewanych o zachorowanie na schozofrenię. Jednocześnie należy zwrócić uwagę, że jej wartość przekracza znacznie średnią dla grupy chorych (tabela 6.3), a dla drugiego podejrzanego pacjenta jest zbliżona do wartości średniej.

Wartości średnie dla osób chorych i zdrowych różnią się znacząco dopiero dla grupy, z której wykluczono dwie osoby o najwyższych częstotliwosciach występowania sakkad. Różnica jest na tyle duża, że możnaby tą metodę wykorzystać do selekcji osób podejrzanych o wystąpienie w przyszłości schizofrenii. Większość policzonych sakkad została wykryta automatycznie przez oprogramowanie „UniJazz". Tylko jedna osoba chora (D3) miała wyniki zbliżone do wartości średniej dla drugiej grupy zdrowej.

Osoba	**SWJ**	**SAC**	**Czas [s]**	**F SWJ** $[\frac{n}{s}]$	**F SAC** $[\frac{n}{s}]$
D1	0	77	28.9	0.00	2.66
D2	4	75	26.3	0.15	2.85
D3	5	52	27.8	0.18	1.87
D4	0	69	26.7	0.00	2.58
P1	0	71	24.9	0.00	2.85
P2	0	30	12.4	0.00	2.42
A1	1	49	22,8	0,04	2,15
A2	1	48	26,7	0,04	1,80
B1	1	35	25,1	0,04	1,39
B2	0	43	26,3	0,00	1,63
C1	0	41	21,6	0,00	1,90
C2	2	66	25,2	0,08	2,62
E1	2	81	24,6	0,08	3,29
E2	2	59	27,3	0,07	2,16
E3	1	33	25,6	0,04	1,29
E4	1	38	22,1	0,05	1,72

Tabela 6.2: Częstotliwość występowania anomalii dla osób chorych zdiagnozowanych (D) i podejrzanych (P) oraz dla osób zdrowych (pozostali)

Test	**Grupa D**	**P1**	**P2**	**Grupa I**	**Grupa II**
SWJ	0.80±0.80	0.00	0.00	0.40±0.30	0.30±0.20
Sakkady	2.49±0.37	2.85	2.42	2.00±0.57	1.76±0.30

Tabela 6.3: Wartości średnie i odchylenia standardowe częstotliwości występowania anomalii dla grupy chorych (Grupa D), wszystkich osób badanych zdrowych (Grupa I), grupy osób zdrowych z wyłączeniem E1 oraz C2 (Grupa II) oraz dla obydwu osób podejrzanych o zachorowanie.

6.3 Podsumowanie

W wyniku niniejszej pracy magisterskiej powstało oprogramowanie „UniJazz", które pozwala na projektowanie i wykonywanie eksperymentów okoruchowych z wykorzystaniem multisensora „Jazz". Funckjonalność wbudowanego w system języka skryptowego pozwoliła na zaprojektowanie kilkunastu podstawowych eksperymentów. Umożliwiają one zapoznanie się z najważniejszymi typami ruchów oka, dzięki czemu system posiada walory edukacyjne. Oprogramowanie mogłoby stać się podstawowym zintegrowanym narzędziem pozwalającym na współpracę z multisensorem „Jazz" znajdującym się w uczelnianym laboratorium okoruchowym.

Jednocześnie udało się wykazać, że za pomocą tego oprogramowania można w prosty i szybki sposób zaprojektować i przeprowadzić badanie eksperymentalne na większej liczbie osób, a następnie przeprowadzić analizę uzyskanych danych. Biorąc pod uwagę dostępność komputerów osobistych (w tym również sprzętu przenośnego) oraz niewielkie rozmiary samego urządzenia rejestrującego ruch oka możliwa jest niezwykła mobilność takiego zestawu. Łatwość w użytkowaniu oprogramowania jak i samego multisensora jest dodatkową zaletą.

Oprogramowanie „UniJazz" zostało zaimplementowane i zaprojektowane w taki sposób, by umożliwić jego dalszy rozwój. Dotyczy to przede wszystkim elemetów języka skryptowego. W szczególności modyfikacja albo dodanie nowego pobudzenia wzrokowego (na przykład umożliwiającego wyprowadzenie na ekran sformatowanego tekstu) jest możliwe bez modyfikacji samego kompilatora. Dzięki zastosowaniu biblioteki `SDL` nie powinno być trudnym utworzenie odrębnej klasy pobudzeń pozwalających na generowanie dźwięków. Rozwój samego oprogramowania dotyczy również sposobów analizy danych. Możliwe jest proste utworzenie dodatkowych filtrów i algorytmów działających na sygnale ruchu oka. Interesującą i niezwykle użyteczną funkcją byłoby pokazywanie pozycji pobudzeń wraz z sygnałem ruchu oka. Należałoby również pomyśleć o utworzeniu specjalnych wykresów i okienek umożliwiających pokazanie sumarycznego ruchu oka dla całego eksperymentu czy nawet odtworzenie jego przebiegu z naniesionym wskaźnikiem pozycji oka.

Spis tabel

Spis rysunków

Bibliografia

[1] Cz. Basztura. *Źródła, sygnały i obrazy akustyczne*. Wydawnictwa Komunikacji i Łączności, Warszawa, 1988.

[2] G. M. Gauthier, J. L. Semmlow, J. L. Vercher, C. Pedrono, and G. Obrecht. Short–Term and Long–Term Adaptative Changes in Eye–Head Movement Coordination Resulting from Reduced Peripheral Vision. In G. Obrecht and W. S. Lawrence, editors, *Presbbyopia Research*. Plenum Press, New York and London, 1992.

[3] H. Kimmig, K. Hauß man, T. Mergner, and C. H. Lücking. What is pathological with gaze shift fragmentation in Parkinson's disease? In *J Neurol*, pages 683–692. 2002.

[4] R. C. Leinecker and Archer T. *Visual C++ Vademecum profesjonalisty*. Wydawnictwo Helion, 2000.

[5] M. R. MacAskill, T. J. Anderson, and J. D. Jonex. Adaptive modification of saccade amplitude in Parkinson's disease. In *Brain*, pages 1570–1582. Plenum Press, New York and London, 2002.

[6] M. H. Nizankowska. *Podstawy okulistyki*. VOLUMED, Wrocław, 2000.

[7] J. K. Rybakowski and A. Borkowska. Eye movement and neuropsychological studies in first–degree relatives of schizophrenic patients. In *Schizophrenia Research*, pages 105–110. 2002.

[8] J. K. Rybakowski, A. Borkowska, P. M. Czerski, and J. Hauser. Eye movement disturbances in schizophrenia and a polymorphism of catechol-O-methyltransferase gene. In *Psychiatry Research*, pages 49–57. 2002.

[9] B. Stroustrup. *Język C++*. Wydawnictwa Naukowo Techniczne, Warszawa, 1998.

[10] J. Szczechura and J. Terelak. Ruchy oczu. In T Sosnowski and K. Zimmer, editors, *Metody psychofizjologiczne w badaniach psychologicznych*, pages 151–181. Wydawnictwo Naukowe PWN, Warszawa, 1993.