Uniwersytet Mikołaja Kopernika
Wydział Fizyki, Astronomii i Informatyki Stosowanej
Instytut Fizyki

Rafał Linowiecki
nr albumu: 244649

Praca magisterska
na kierunku Informatyka Stosowana
specjalność: Inteligencja obliczeniowa i sieci komputerowe

Kontrola aplikacji za pomocą okulografu (współpraca z ICNT)

Opiekun pracy dyplomowej
dr hab. Jacek Matulewski
Zakład Mechaniki Kwantowej
Wydział Fizyki, Astronomii
i Informatyki Stosowanej

Toruń 2015

Pracę przyjmuję i akceptuję

.....
data i podpis opiekuna pracy

Potwierdzam złożenie pracy dyplomowej

.....
data i podpis pracownika dziekanatu

Dziękuję dr hab. Jackowi Matulewskiemu za poświęcony czas, cenne rady merytoryczne oraz zaangażowanie przy wspólnym tworzeniu platformy GCAF.

Dziękuję dr Bibiannie Bałaj za życzliwość, pomoc oraz cenne wskazówki podczas współpracy.

Dziękuję Rodzicom za okazane wsparcie, cierpliwość oraz wszelką pomoc przy realizacji życiowych celów.

Spis treści

1 Wstęp **6**

2 Okulograf **8**

2.1 Historia okulografu 8

2.2 Budowa i zasada działania okulografu 10

2.3 Stany oka 18

2.4 Zastosowanie okulografów 19

3 Język GIML **25**

3.1 Nowe podejście do tworzenia aplikacji 25

3.2 Struktura GIML 26

3.3 Znacznik `ustawienia` 29

3.4 Znaczniki zasobów 30

3.5 Znaczniki `sceny` i `scena` 33

3.6 Znacznik `obszar` 37

3.7 Przykłady użycia i perspektywy rozwoju 44

4 Platforma GCAF **45**

4.1 Wymagania aplikacji 45

4.2 Opis aplikacji 45

4.3 Okno konfiguracyjne 48

4.3.1 Zakładka *Na skróty* 49

4.3.2 Zakładka *Eyetracker* 50

4.3.3 Zakładka *Rejestrowanie i analiza danych, statystyki* 52

4.3.4 Zakładka *Aplikacja* 54

4.3.5 Zakładki *Rejestr zdarzeń* i *Analiza pozycji oka* 56

4.3.6 Zakładka *Ustawienia* 58

4.3.7 Zakładka *O...* . . . 58
4.4 Działanie programu i sterowanie wzrokiem . . . 59
4.5 Dane zapisywane przez program . . . 62

5 Eksperymenty z użyciem GCAF . . . 64
5.1 Bajka interaktywna . . . 64
5.2 Czerwona kropka . . . 65
5.3 Elmo . . . 66
5.4 Koła . . . 68

6 Podsumowanie . . . 69

7 Bibliografia . . . 69

1. Wstęp

Postęp technologiczny znacząco zmienił nasze codzienne życie, które w wielu aspektach zdominowane jest przez elektronikę. Przygotowanie posiłków wymaga umiejętności obsługi kuchenki mikrofalowej lub piecyka z elektronicznymi wyświetlaczami, pralka – to samo, komunikacja bez obsługi smartphone's staje się niemożliwa. A wszystko dla naszego komfortu. Wszechobecność elektroniki ma jednak także ciemną stronę. Nie wszyscy są wstanie nadążać za ciągle zmieniającymi się urządzeniami np. z powodu ograniczonych zdolności poznawczych związanych z wiekiem (niemowlęta i osoby starsze), chorobą bądź upośledzeniem fizycznym lub psychicznym.

Weźmy dla przykładu zwykły komputer, obecnie nieodzowny w wielu zajęciach. Nie może go użyć osoba, która nie jest w stanie posłużyć się myszą i klawiaturą. Są to osoby niepełnosprawne, szczególnie sparaliżowane, ale także małe dzieci. W przypadkach obu tych grup nie pomogą także nowe i intensywnie rozwijane alternatywne metody bardziej naturalnej interakcji człowieka z komputerem, a więc głos i gest. Pozostają najbardziej podstawowe kanały komunikacji – wzrok i aktywność mózgu (np. EEG). Pierwsza możliwość jest zwykle łatwiejsza w realizacji i na niej się w tej pracy skupię.

Sterowanie komputerem za pomocą wzroku jest możliwe dzięki okulografowi (ang. *eyetracker*), któremu został poświęcony pierwszy rozdział pracy. Daje on zupełnie nowe możliwości korzystania z komputerów zarówno dzieciom, osobom starszym, a przede wszystkim osobom niepełnosprawnym. Ale nie tylko nim. Także zupełnie sprawny użytkownik komputera może zyskać korzystając z eyetrackera. Nowy sposób sterowania w grach, wyświetlanie dodatkowych informacji o elementach interfejsu, na które spojrzał użytkownik, czy analiza ruchu oczu to wybrane zastosowania, których użyteczność jest na razie trudna do oceny, lecz wydaje się bardzo interesująca.

Kolejne rozdziały pracy zostały poświęcone przedstawieniu oprogramowania *Gaze Controlled Applications Framework* (GCAF), czyli platformy do tworzenia aplikacji sterowanych wzrokiem, której część została stworzona w ramach realizacji mojej pracy magisterskiej. W szczególności zajmowałem się przygotowaniem implementacji sterownika do obsługi eyetrackera firmy Mirametrix, gromadzeniem i analizą statystyk obszarów zainteresowania, obsługą dźwięku, filmów, możliwością nagrywania przebiegu aplikacji, częścio-

wym pokryciem kodu źródłowego testami oraz pomocą podczas wdrażania programu w różnych eksperymentach i promowaniu produktu na konferencji, wystąpieniach, konkursach i podczas spotkań z inwestorami. Pomysł stworzenia tej platformy powstał w trakcie prac nad projektem NeuroPerKog realizowanym w Interdyscyplinarnym Centrum Nowoczesnych Technologii UMK w Toruniu, którego kierownikiem jest prof. Włodzisław Duch, a w którym była potrzeba tworzenia aplikacji kontrolowanych wzrokiem przez niemowlęta. Aktualnie, w oparciu o tą platformę przeprowadzane są cztery eksperymenty: bajka interaktywna, czerwona kropka, elmo oraz koła. Istnieją także plany, aby udostępnić lub ewentualnie skomercjalizować platformę GCAF z myślą o znacznym zmniejszeniu kosztów przygotowywania spersonalizowanego oprogramowania dla osób niepełnosprawnych.

2. Okulograf

Okulograf (łac. *oculus* - oko) jest urządzeniem, które śledzi ruch gałek ocznych i potrafi wskazać punkt widzenia (np. na ekranie monitora). Obecnie częściej używana jest angielska nazwa okulografu, czyli *eyetracker*. Dziedzina nauki, która zajmowała się zagadnieniem śledzenia wzroku jest nazywana *okulografią* (ang. *eyetracking*). Rozdział ten poświęcony został opisowi eyetrackera, jego historii, zasadzie działania oraz możliwości, które ze sobą niesie w codziennym życiu.

2.1 Historia okulografu

Śledzenie wzroku jest naturalnym sposobem komunikacji. Niemowlę odruchowo podąża za wzrokiem rodzica, a rodzic dziecka. Ta interakcja umożliwia naukę, w szczególności naukę języka. Pierwsze metody śledzenia wzroku używano do analizy sposobu czytania tekstu przez osoby badane. Interesowano się również odczuciami i zamierzeniami osób, czego poznanie było możliwe m.in. na podstawie obserwacji ruchu wzroku.

Pierwszy okulograf powstał na przełomie XVIII i XIX wieku. Ich konstrukcja nie była skomplikowana, ale była uciążliwa dla pacjentów, ze względu na mechaniczne elementy, które miały kontakt z powierzchnią gałki ocznej. Tego typu okulografy są nazywane *inwazyjnymi*. Za pioniera okulografii[1] uważa się Louisa Javala, francuskiego okulistę, który w swoich badaniach w latach 1878-79 po raz pierwszy wykorzystał okulograf inwazyjny [9, zakł. historia]. Był to pierwszy krok do obiektywnego pomiaru ruchu oka.

W roku 1898 Edmund Burke Huey wymyślił sposób na unieruchomienie głowy pacjenta poprzez specjalną podstawkę z zamontowanym do zagryzienia ustnikiem z częściowo schłodzonym woskiem. Również w tym samym roku Edmund Delabarre zaproponował, aby znieczulić gałkę oczną podając roztwór z zawartością 2-3% kokainy, oczywiście nadal z mechanicznym urządzeniem przyłożonym do oka. Ważnym odkryciem w dziedzinie okulografii było urządzenie Raymonda Dodge'a i Thomasa Sparksa Cline'a z roku 1901, które rejestrowało na światłoczułej płytce światło odbite od poruszającej się gałki ocznej (bez dodatkowych przyczepianych elementów). Niestety urządzeniem rejestrowane

[1] Okulografia - nauka zajmująca się śledzeniem wzroku przy użyciu okulografu, obecnie eyetrackera

były wyłącznie poziome ruchy oka [9, zakł. historia]. Sposób ten dał początek okulografom *bezinwazyjnym*, które są obecnie najpopularniejsze[1, s.9]. Warto tutaj zaznaczyć, że oprogramowanie GCAF wykorzystuje do śledzenia oka eyetrackery właśnie tego typu.

Okulografy inwazyjne, które potrafiły rejestrować ruch oka w dwóch płaszczyznach (tj. w pionie i w poziomie) skonstruowano w 1905 roku. Jego twórcami byli Charles Judd, Matt McAlister oraz William Gladstone Steel. Wówczas zmiany położenia oka były rejestrowane na światłoczułej taśmie za pomocą promienia odbitego od znacznika mechanicznego, który był mocowany do oka osoby badnej [9, zakł. historia].

Głównym problemem pierwszych okulografów była konieczność unieruchomienia głowy osoby badanej. W przypadku inwazyjnych mechanicznych urządzeń było to szczególnie ważne ze względów bezpieczeństwa i zdrowia pacjentów. Przełom nastąpił w roku 1948, w którym H. Hertridge i L.C. Thompson skonstruowali okulograf w formie opaski zakładanej na głowę [9, zakł. historia]. Jego użycie nie wymagało unieruchamiania głowy badanego, co było wielką zaletą tego urządzenia. Obecnie bardzo popularne są eyetrackery tego typu, które potocznie nazywa się *nagłownymi*.

Od lat 50-tych XX wieku indywidualne ośrodki badawcze opracowały szereg różnych technik służących rejestracji ruchu oka. W latach 1950 - 1970 używane przez m.in. A. Yarbusa i R. W. Ditchburna systemy soczewek kontaktowych z lustrami były przykładem urządzeń dokonujących bardzo precyzyjnych pomiarów, jednak ich używanie było bardzo niewygodne. Innym podejściem było zastosowanie elektromagnetyzmu, a dokładniej systemów cewek elektromagnetycznych, które mierzyły indukcję elektromagnetyczną w krzemowej soczewce kontaktowej umieszczonej na znieczulonym oku. Taki system był przez długi czas uważany za najbardziej precyzyjną metodę pomiaru dowolnych ruchów oka (Collewijn - rok 1998). Problemem tego typu urządzeń była konieczność indywidualnego dopasowania soczewek do osoby badanej. Jednak czasami nawet to nie poprawiało komfortu ich używania. Podobnym pomysłem był elektrookulograf, znany jako EOG. Jest to system pomiaru magnetycznej zmienności dipoli z mięśni gałki ocznej. Wadą takiego rozwiązania jest horyzontalny pomiar ruchów oka (łatwy dostęp do mięśni poruszających oko w poziomie, trudny w przypadku odczytu impulsów mięśni odpowiadających za ruch pionowy) oraz duża podatność na zakłócenia elektromagnetyczne z otaczających mięśni. Tego typu urządzenia nadal są stosowane ze względu na wysoką częstotliwość próbkowa-

nia[2] [1, s. 10].

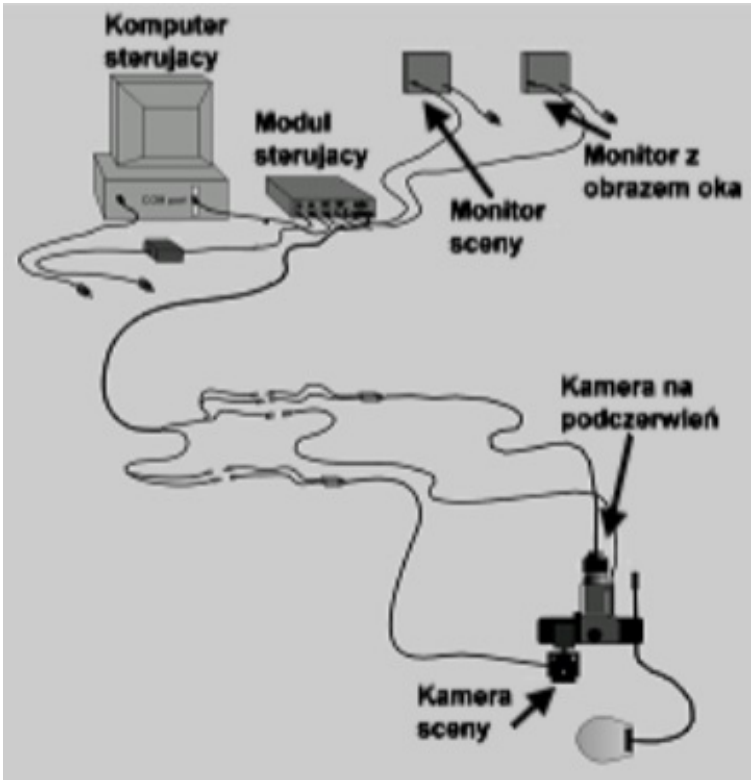
Obecny rynek okulografów jest bardzo zróżnicowany. Do każdego rodzaju badania można dopasować oferowany przez różne firmy okulograf, który będzie wykazywał się ustalonym poziomem precyzji oraz odpowiednim kosztem zakupu. W ostatnim czasie duży nacisk położono na stworzenie "tanich" eyetrackerów, których niższy koszt okupiony jest jednak znacznie gorszymi parametrami. Są one kierowane głównie do indywidualnych klientów chcących używać okulografu np. do sterowania komputerem. Uzyskana w ich przypadku precyzja jest wystarczająca, aby swobodnie korzystać z takiego narzędzia jako kontrolera. Popularne stają się również okulografy w formie okularów. Są proste w obsłudze i można z nich korzystać przez dłuższy czas, jednak ich precyzja jest niższa niż w przypadku okulografów stacjonarnych, przy porównywalnej cenie. W ich przypadku zyskujemy jednak mobilność, co pozwala np. na badanie bez konieczności pozostawania przed monitorem. Ze strony specjalistów z branży eyetrackingowej oraz nowych technologii pojawiają się spekulacje na temat upowszechnienia używania eyetrackerów, m.in. w smartfonach i tabletach. Nie ma jednak precyzyjnych informacji, w jaki sposób konstruowane byłyby eyetrackery dla tego typu urządzeń.

2.2 Budowa i zasada działania okulografu

Głównymi czynnikami, które wpływają na poprawność pracy eyetrackera jest szybkość i dokładność użytej do pomiaru kamery oraz poprawne działanie przy różnym oświetleniu [2, s. 326]. Wiele oferowanych dzisiaj urządzeń nie posiada szczegółowych informacji o parametrach poszczególnych podzespołów. O ile w przypadku indywidualnego klienta problem ten na ogół nie występuje, ponieważ nie jest wymagana duża precyzyjność okulografu lub różnice w działaniu trudno dostrzec zwykłemu użytkownikowi o tyle w przypadku precyzyjnych rozwiązań (np. eksperymenty naukowe) każdy parametr będzie miał istotny wpływ na otrzymywane wyniki. Pojawia się więc potrzeba, aby produkowane okulografy cechowały się wydajnością i precyzyjnością przy stosunkowo małym koszcie ale z pełną specyfikacją sprzętową.

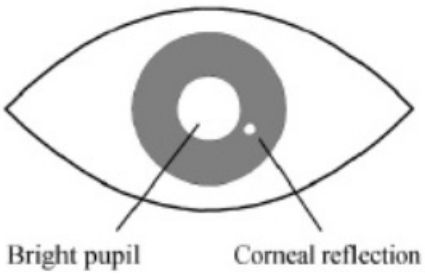
Przykładem niezbyt skomplikowanego eyetrackera może być system przedstawiony na rys. 2.1. System ten służy do monitorowania położenia gałek ocznych przy użyciu kompu-

[2]Częstotliwość próbkowania - ilość próbek sygnału w jednej jednostce czasu



Rys. 2.1 Schemat systemu nagłownego do śledzenia ruchów oczu. Źródło: [5, s.2].

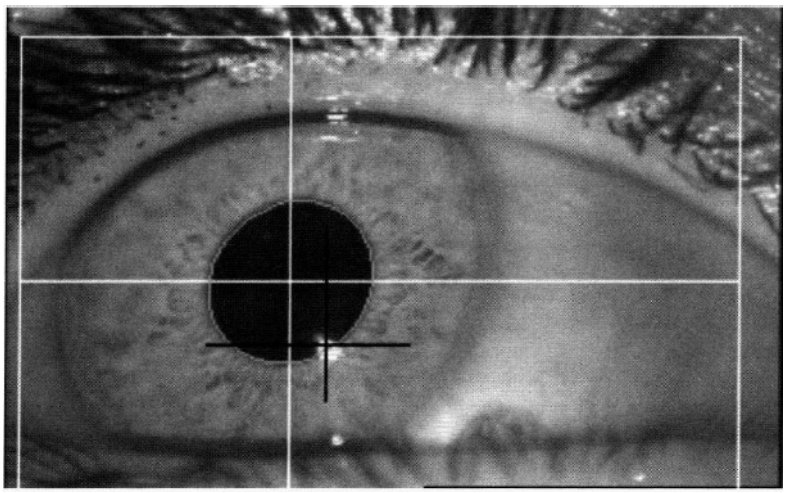
tera oraz dwóch kamer. Jedna z nich - kamera sceny rejestruje obraz, który widzi osoba badana. Druga z kamer (kamera na podczerwień) jest montowana na opasce i śledzi ruch oka. Dwa monitory służą do podglądu w czasie rzeczywistym obrazów z poszczególnych kamer. Całym procesem steruje komputer z odpowiednim oprogramowaniem, który zbiera informację z obu kamer [5, s.3]. Wektor położenia wzroku jest obliczany pomiędzy źrenicą a refleksem rogówki w celu uwzględnienia wszelkich ruchów głowy [8, s.10]. Obrazuje to rys. 2.2. Źrenica jest jasna z powodu światła podczerwonego odbitego od siatkówki



Rys. 2.2 Refleks rogówki i oświetlenie źrenicy w przechwyconym obrazie kamery IR. Źródło: [8, s.10].

poprzez źrenicę. System odbić źrenica-rogówka właściwie identyfikuje źrenicę, jako skrzy-

żowanie białych linii oraz refleks rogówki, jako skrzyżowanie czarnych linii na obrazie oka [4, s. 25]. Ilustruje to rys. 2.3.

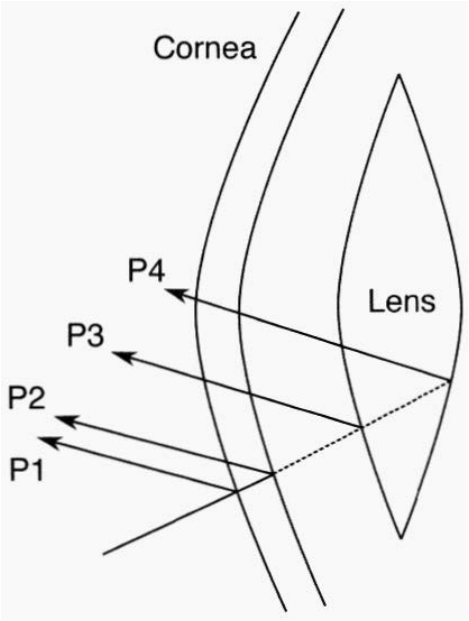


Rys. 2.3 Schemat rozpoznawania źrenicy i refleksu rogówki na obrazie oka z kamery IR. Źródło: [4, s.25].

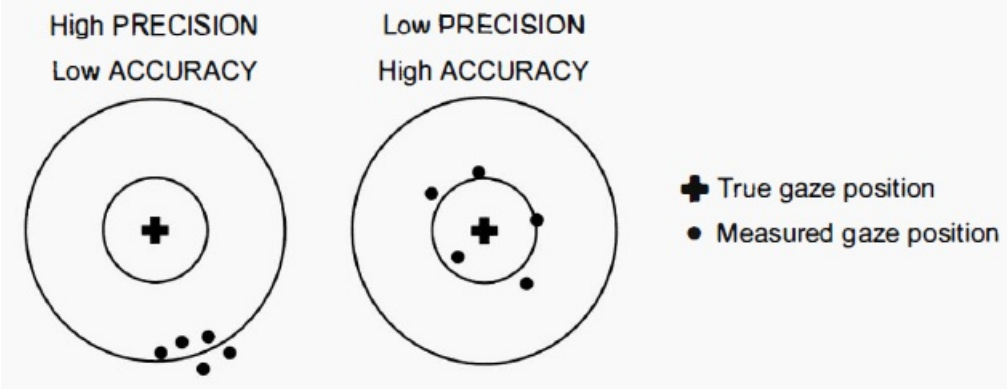
Refleksy rogówki są znane jako *refleksy Purkinjego* lub *obrazowanie Purkinjego*. Ze względu na budowę oka, wyróżniane są cztery refleksy Purkinjego, które przedstawia rys. 2.4. Pierwszy z nich (P1) jest odbiciem od przedniej powierzchni rogówki. Drugi (P2) to odbicie od tylnej powierzchni rogówki. Trzecim refleksem (P3) jest odbicie od przedniej powierzchni soczewki. Czwarty refleks (P4) to odbicie od tylnej powierzchni - prawie tej samej wielkości i tworzy się w tej samej płaszczyźnie co pierwszy refleks Purkinjego ale ze względu na zmiany w wartości współczynnika załamania światła, z tyłu soczewki intensywność jest mniejsza o 1% w odniesieniu do pierwszego refleksu.

Przy pomiarze wektora położenia wzroku istotnymi parametrami są dokładność oraz precyzja działania okulografu. Często te dwa pojęcia są mylone. Dokładnością eyetrackera jest (średnia) odległość pomiędzy rzeczywistą pozycją wzroku a pozycją zarejestrowaną przez urządzenie. Precyzja natomiast jest zdefiniowana, jako zdolność eyetrackera do wiarygodnego odtworzenia pomiaru. Różnicę pomiędzy dokładnością a precyzją ilustruje rys. 2.5. Oczywiście dobre eyetrackery powinny posiadać zarówno wysoką dokładność, jak i wysoką precyzję [4, s.33].

Podstawowym elementem obecnych okulografów jest sensor obrazu, będący główną częścią kamery. Stąd czasem mylnie sam sensor obrazu jest utożsamiany z pojęciem kamery. Jego głównym zadaniem jest konwersja danych optycznych na sygnał elektryczny.



Rys. 2.4 Cztery refleksy Purkinjego. Źródło: [4, s.22].



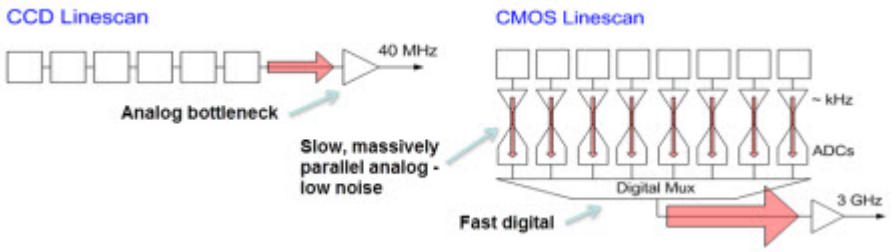
Rys. 2.5 Dokładność a precyzja. Źródło: [4, s.34].

Czynnikiem, który decyduje o jego wydajności jest materiał, z którego został wyprodukowany. W przypadku okulografów sensory te robi się głównie z krzemu (są wrażliwe na fotony w świetle widzialnym i bliskim podczerwoności), jednak niektóre sensory służące do rejestracji fal podczerwonych są produkowane z półprzewodników np. z antymoneku indu (InSb) lub arsenka indu i galu (InGaAs). Sensor jest złożony z macierzy pikseli. Aby odczytać obraz z sensora, korzysta się z jednej z dwóch dostępnych metod skanowania obrazu. Pierwszą metodą jest skanowanie progresywne, które polega na odczycie kolejno wszystkich pikseli. Dzięki temu można uzyskać wysoką jakość obrazu. Drugi rodzaj to skanowanie z przeplotem, które polega na odczycie co drugiego wiersza macierzy pikseli. Takie podejście zmniejsza ilość informacji potrzebną do stworzenia obrazu, jednak jego

jakość jest gorsza. W dodatku pojawia się charakterystyczne migotanie obrazu. Aby temu zapobiec często stosuje się przeplot podwójny, czyli osobny przeplot dla wierszy parzystych i nieparzystych macierzy. Później uzyskany obraz łączy się w jedną ramkę. Taki zabieg poprawia dwukrotnie rozdzielczość przestrzenną, jednak dwukrotnie zmniejsza rozdzielczość czasową takiego urządzenia [2, s. 327]. Z tego względu wskazane jest użycie skanowania progresywnego, co obecnie jest stosowane w nowoczesnych urządzeniach przetwarzających obraz.

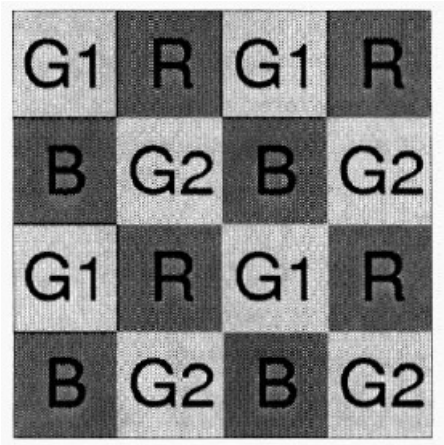
Mówiąc o sensorach obrazu trudno nie wspomnieć o dwóch popularnych typach sensorów, mianowicie CCD (Charge-Coupled Device) i CMOS (Complementary Metl-Oxide-Semiconductor). Pierwszy z nich to płytka krzemowa, która jest zbudowana ze światłoczułych elementów. Oczywiście na płytce tej znajdują się piksele tworząc macierz pikseli. Im większa rozdzielczość matrycy (przy zachowaniu jej fizycznych rozmiarów) tym mniejszy rozmiar każdego z pikseli. W przypadku CCD każdy z pikseli posiada dołączoną elektrodę. Przyłożenie napięcia powoduje powstanie studni potencjału. Kiedy rejestrowany foton pada na płytkę, zostaje wybity elektron na danej elektrodzie, który zostaje uwięziony w powstałej studni potencjału. Proces "zbierania" elektronów z powierzchni jest nazywany *ekspozycją* lub potocznie zbieraniem elektronów. Po tym procesie następuje przetworzenie ilości uwięzionych elektronów, uzyskując informację o natężeniu padających na płytkę fotonów. Kolejny etap to przetwornik analogowo-cyfrowy, który przetwarza informację o padających fotonach na impulsy elektryczne interpretowane przez urządzenia wyświetlające obraz. Problemem w tego typu sensorach jest szybkie nagrzewanie się układu, co powoduje wprowadzanie dodatkowych zakłóceń do przesyłanych informacji o fotonach [10, zakł. CCD]. W przypadku sensorów CMOS są one zbudowane z układów tranzystorów (MOS), które składają się z trzech warstw: płytki wyciętej z monokryształu krzemu, na którą napyla się izolator (krzemionkę), po czym napylona zostaje cienka warstwa przewodnika - np. złota. Główną zasadą tej technologi jest fakt, że tranzystory te ustawia się w taki sposób, aby przy różnym stanie logicznym w danej chwili tylko jeden z nich przewodził. Główną zaletą tego rozwiązania jest możliwość jednoczesnego odczytu całej macierzy pikseli, czego nie można dokonać w sensorach CCD. Obecnie sensory CMOS znajdują coraz szersze zastosowanie - m.in. w budowie procesorów [10, zakł. CMOS]. Ponieważ w systemie CMOS każdy z pikseli posiada własny przetwornik analogowo-cyfrowy, prędkość przesyłania informacji jest kilkukrotnie większa, niż w przypadku technologii

CCD. Ukazuje to rys. 2.6. Również zużycie energii jest mniejsze, ponieważ nie ma konieczności ciągłego utrzymywania napięcia układu.



Rys. 2.6 Porównanie wydajności sensorów typu CCD oraz CMOS. Źródło: [11].

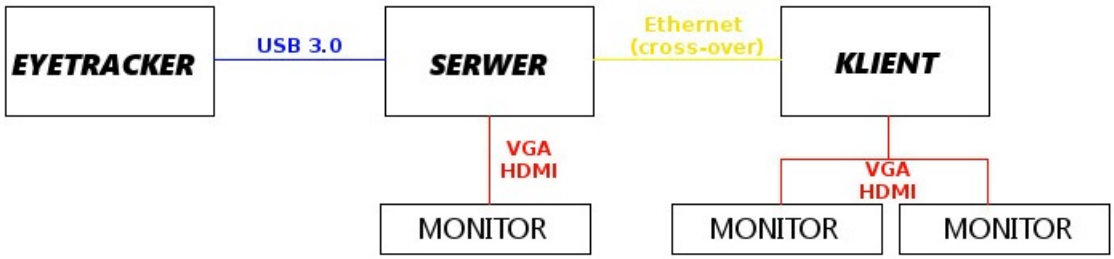
Pozostaje pytanie odnośnie rejestrowania kolorów przez sensory optyczne. W tym celu korzysta się z tzw. czujników koloru zawierających mikrofiltry trzech różnych typów. Każdy z nich posiada maksymalną czułość na konkretne regiony długości fali (czerwony, zielony lub niebieski). Wszystkie trzy filtry połączone w jeden obiekt są nazywane *filtrem Bayera* (rys. 2.7) [2, s. 327].



Rys. 2.7 Konstrukcja filtru Bayera. Źródło: [2, s.328].

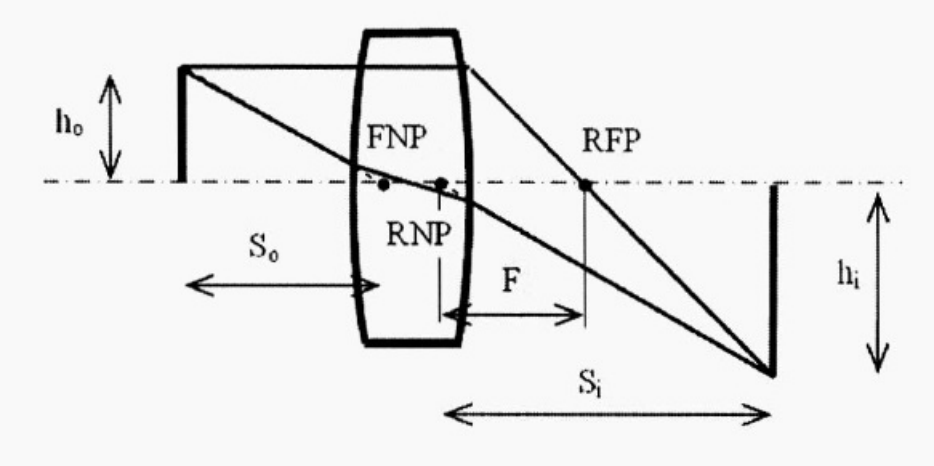
Kolejnym istotnym elementem w budowie okulografu jest interfejs przesyłu danych pomiędzy eyetrackerem a komputerem. Dawniej wykorzystywano łącza analogowe - obecnie cyfrowe. Obecne okulografy są zazwyczaj wyposażone w porty USB oraz gniazda Ethernet. Port USB ze względu na swoją uniwersalność wyparł dawniej używany port typu RS232, jednak w starszych modelach okulografów można spotkać jeszcze tego typu rozwiązanie. Port Ethernet (gniazdo RJ-45) pozwala na przesył informacji przy użyciu kabla sieciowego, co daje możliwość podłączenia okulografu do sieci. Ten rodzaj połączenia jest bardzo ważny z punku widzenia budowania oprogramowania do obsługi okulografu.

Sterowniki odpowiedzialne za komunikację z okulografem bazują na poleceniach wysyłanych protokołem TCP/IP właśnie przez połączenie sieciowe. Schemat wykorzystywanych interfejsów przesyłu danych dla platformy GCAF został przedstawiony na rys. 2.8.



Rys. 2.8 Schemat łączy wykorzystywanych dla GCAF

System optyczny przedstawiony na rys. 2.9 to kolejny element struktury wewnętrznej okulografu. W jego skład wchodzą soczewki optyczne, których zadaniem jest uzyskanie ostrego obrazu oka z bliskiej odległości. Soczewki te są złożonymi systemami, na które składają się soczewki proste oraz mechaniczne lub elektromechaniczne mechanizmy do ich sterowania. Głównymi parametrami charakteryzującymi soczewki są: długość ogniskowej



Rys. 2.9 Formowanie obrazu i główne punkty soczewki. Źródło: [2, s. 332]

(F), liczba-f oraz typ adaptacji. Długość ogniskowej (F) można przedstawić jako parametr, który informuje, jak bardzo soczewka powiększa obiekty. Określa on w jakiej odległości od eyetrackera powinien znajdować się użytkownik, aby uzyskać wyraźny obraz. Warto tutaj wspomnieć o tylnym punkcie ogniskowej oraz przednich i tylnych punktach węzłowych. Tylny punkt ogniskowej (ozn. RFP) jest miejscem, gdzie koncentrują się promienie

równoległe do osi układu optycznego. Przednie i tylne punty węzłowe (ozn. odpowiednio FNP i RNP) mają tę własność, że promień, który przechodzi przez jeden z nich zostaje załamany przez soczewkę w taki sposób, że wydaje się, jakby pochodził z drugiego punktu i miał ten sam kąt w stosunku do osi optycznej. Długość ogniskowej jest określana, jako odległość pomiędzy tylnym punktem węzłowym a tylnym punktem ogniskowej i wyraża się zależnością:

$$\frac{1}{S_0} + \frac{1}{S_1} = \frac{1}{F} \tag{2.1}$$

gdzie S_0 jest odległością pomiędzy obiektem a przednim punktem węzłowym i S_1 jest odległością pomiędzy tylnym punktem ogniskowej a płaszczyzną, w której obraz jest najlepiej skoncentrowany [2, s. 331, 332].

Ważnym parametrem przy układach soczewek jest przesłona, która zależy od otwarcia soczewki i opisuje stopień kolimacji[3] promieni świetlnych wpadających do obiektywu. Wyraźny obraz uzyskuje się, gdy otwarcie jest małe. Gdy otwarcie jest większe obraz staje się bardziej niewyraźny. Często przesłona jest charakteryzowana liczbą-f (FN). Liczba-f jest funkcją długości ogniskowej i średnicy otwarcia soczewki (D_{op}):

$$FN = \frac{F}{D_{op}} \tag{2.2}$$

Większe otwory charakteryzują się mniejszą liczbą-f. Soczewki z większymi otworami są określane "szybszymi", ponieważ czas migawki może być szybszy dla takiego samego stopnia naświetlenia. Mniejsza wartość szczeliny oznacza, że obiekty mogą znajdować się w szerszym zakresie odległości, co nazywane jest *głębokością pola*. Większy parametr f w przypadku okulografów oznacza większą głębie pola. Oddalenie się od kamery powoduje jednak, że powierzchnia, z której pobierane jest światło staje się mniejsza, a wówczas należy zastosować silniejsze oświetlenie oka [2, s. 332].

Większość okulografów dostępnych na rynku używa światła podczerwonego, najczęściej pochodzącego z diod LED. Kamery są przystosowane do przechwytywania różnych długości fal świetlnych. Dodatkowo stosuje się filtry, które tłumią te zakresy długości fal, których źródłem nie jest światło eyetrackera odbite od oka.

Dużym przełomem w poprawianiu wydajności okulografów mogą okazać się nowe dynamiczne czujniki wizyjne, które są próbą stworzenia nowego typu sensora optycznego. Główna różnica polega na tym, że zamiast przesyłać duże ilość niepotrzebnych danych w

[3]Kolimacja - to proces przetwarzania rozbieżnych wiązek promieniowania na wiązki równoległe.

postaci pełnych obrazów klatka po klatce, przesyłane będą tylko zmiany dotyczące pojedynczych pikseli. To oznacza, że informacje będą przesyłane tylko, jeśli któryś z pikseli zarejestruje zmianę obserwowanego obszaru. Oczywistą konsekwencją zastosowania tego podejścia jest znaczny wzrost zarówno rozdzielczości czasowej, jak również wzrost przepustowości. Obecne badania wskazują, że wzrost wydajności może być ponad dwukrotna w odniesieniu do aktualnie stosowanych czujników[2, s. 334].

2.3 Stany oka

Pierwszym rozróżnialnym stanem oka są *sakkady*. Nazywany jest w ten sposób ruch gałki ocznej zmieniający punkt widzenia do nowego obszaru, który jest oddalony o więcej niż 2 stopnie kątowe. Po sakkadzie następuje moment, w którym oko pozostaje w bezruchu w celu zarejestrowania bodźca wzrokowego. Taki stan, który trwa ponad 100 ms jest nazywany *fiksacją*. Innym stanem są *mikrosakkady*, które są nieregularnymi ruchami pojawiającymi się podczas trwania fiksacji. W przypadku, kiedy cel się porusza, wzrok płynie podąża za nim. To tzw. *płynne prowadzenie wzroku* [3, s. 17,18]. Ciekawostką jest, że człowiek nie potrafi w płynny sposób poruszać oczami bez prowadzonego wzrokiem obiektu. Często analizie są poddawane również *mrugnięcia*, w szczególności podczas badań reakcji emocjonalnych badanego. Oprócz tych wymienionych stanów oka, w badaniach naukowych często rejestruje się również inne dane. Wszystkie stany zostały przedstawione w tab. 2.1.

Tablica 2.1 Typy stanów oka. Źródło: [2, s. 18]

Typ stanu oka	**Opis**
Fiksacja	Stan bezruchu > 100 ms z 1 stopniem kątowym widzenia
Sakkada	Ruch na nowe obszary ze zmianą polem widzenia o więcej niż 2 stopnie
Mikrosakkada	Bardzo małe ruchy, które występują nieregularnie podczas fiksacji
Płynne prowadzenie	Ruch oka podążającym za obiektem z taką samą prędkością i torem jak obiekt
Konwergencja	Wewnętrzny ruch oczu, np. w przeciwnych kierunkach, do uzyskania lepszej ostrości siatkówki dla bliskich obiektów
Odruch przedsionkowo-oczny	Ruch w celu utrzymania punktu w polu widzenia podczas ruchu głowy
Odruch optokinetyczny	Gładkie sakkadowe ruchy podążające za ruchomymi scenami
Przystosowanie	Zmiany w kształcie soczewki do skupiania światła odbitych od przedmiotów znajdujących się w różnych odległościach
Rozszerzanie się źrenicy	Zmiana wielkości otworu w tęczówce aby utrzymać optymalny poziom światła wewnątrz oka

2.4 Zastosowanie okulografów

Eyetrackery dopiero wchodzą na rynek urządzeń niemedycznych. Istotne jest więc, do jakich celów mogą być użyte. Im większe pole użycia produktu, tym jego popularność na rynku rośnie a cena maleje. Taka sytuacja ma miejsce właśnie w przypadku rynku okulografów. Podstawą tutaj jest dostarczenie informacji na temat rozróżnienia i przetworzenia różnych stanów oka w określonych sytuacjach.

Badania okulograficzne w ostatnich latach zyskały coraz szersze zastosowanie w badaniach dotyczących motoryzacji. Dokonuje się analizy zachowań kierowców zarówno w re-

alnych warunkach drogowych, jak również z wykorzystaniem różnych symulatorów. Głównym celem prowadzenia tego typu badań jest poprawa warunków panujących na drogach. Analizuje się wpływ oznakowania drogi, elementów infrastruktury drogowej, a także innych obiektów znajdujących się w zasięgu pola widzenia kierowcy na poziom jego koncentracji oraz poziom widoczności poszczególnych obiektów. Wykorzystanie okulografów pozwala dokonywać analizy dróg istniejących oraz dróg wprowadzanych do symulatora. W pierwszym przypadku stosuje się dodatkowo specjalny system kamer umieszczonych w pojeździe, który rejestruje przebieg drogi z punktu widzenia kierowcy. Na ten obraz można wówczas nałożyć punkt wzroku, żeby analizować, na które obiekty patrzył kierowca. Ponadto nagrany obraz drogi zostaje później przetworzony i umieszczony w symulatorze jako obszar, w którym będzie prowadzona symulacja. Wyniki badań okulograficznych są wykorzystywane do oceny bezpieczeństwa infrastruktury drogowej w aspekcie widoczności kluczowych elementów drogi, co może zostać wykorzystane również na etapie projektowania nowych dróg. Tego typu informacje są szczególnie istotne podczas audytu[4] BRD[5] [6, s. 123].

Jednak zastosowanie okulografii w motoryzacji nie kończy się tylko na analizie infrastruktury drogowej. W ten sposób pozyskuje się również informacje na temat budowy przyrządów kontrolno-pomiarowych i wskaźników samochodu, z których korzystają kierowcy. Podobnie w przypadku samolotów. Informacje te są wykorzystywane głównie do poprawienia ergonomii rozmieszczenia tych elementów [9, zakł. zastosowania]. Głównym celem jest w tym przypadku możliwość obsługi urządzeń pojazdu, bez konieczności odrywania wzroku kierowcy od drogi. Aktualnie prowadzone są również badania przez czołowych producentów samochodów, które dotyczą poziomu oświetlenia drogi, w związku z wprowadzaniem nowych technologii źródła światła w reflektorach samochodowych. Oferowane reflektory złożone z kilkunastu diod LED oraz reflektory laserowe, które już pojawiły się na rynku, są badane pod kątem oświetlenia pola widzenia kierowcy w różnych sytuacjach drogowych. Wiązka światła jest sterowana w pełni automatycznie poprzez stosowanie elementów światłoczułych, które natychmiastowo reagują (np. na nadjeżdżający z przeciwka pojazd) i zmieniają intensywność strumienia oraz kształt wysyłanej z reflektorów wiązki światła tak, aby uzyskać optymalny stopień oświetlenia drogi przy zachowaniu maksymalnego bezpieczeństwa zarówno kierowcy, jak i osób znajdujących się

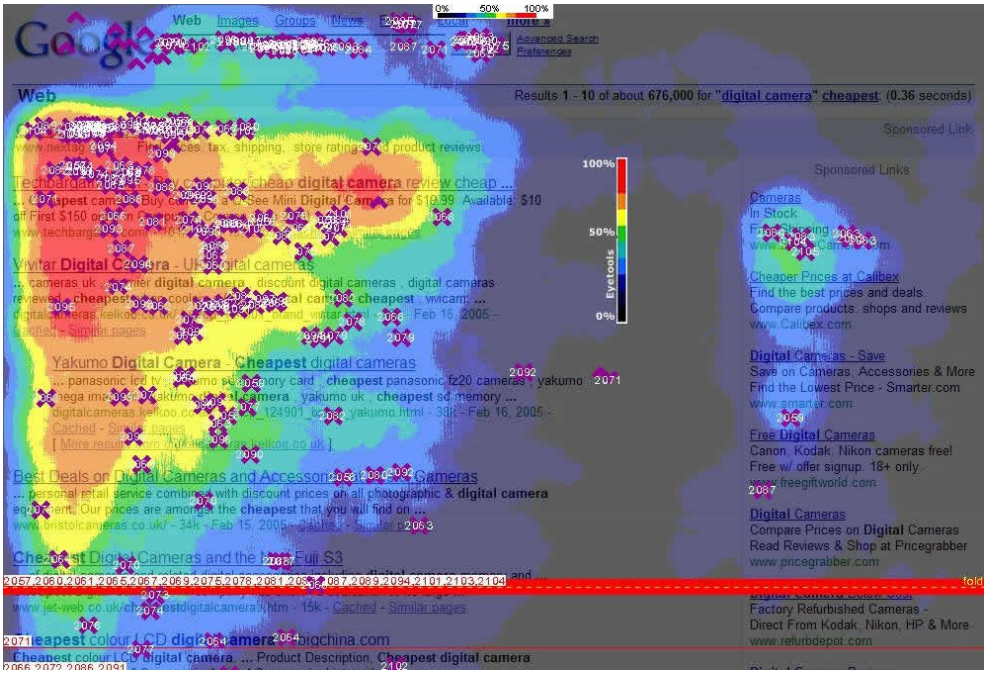
[4]Audyt - niezależna ocena danego systemu pod względem zgodności z określonymi normami.

[5]BRD - bezpieczeństwo ruchu drogowego.

poza pojazdem.

Także piloci są testerami w symulatorach lotu z użyciem okulografów. W większości przypadków stosuje się okulografy mobilne w formie okularów, dzięki czemu pilot nie czuje dyskomfortu podczas symulacji a sam okulograf w żaden sposób nie ogranicza jego ruchów. Podobne symulacje przeprowadza się w pojazdach wojskowych, badając w ten sposób ich jakość użytkową, jak również sposób budowy poszczególnych elementów kokpitów. Ważnym zastosowaniem okulografów w kontekście wojska jest użycie wzroku do sterowania urządzeniami naprowadzającymi [9, zakł. zastosowania].

Okulografy mają również swoje zastosowanie w badaniach marketingowych, szczególnie do walidacji reklam lub stron internetowych. Okulogrf pozwala na ocenę zainteresowania poszczególnymi elementami umieszczonymi na stronie lub reklamie. Rejestrowany wzrok osoby badanej jest następnie przetworzony np. do postaci mapy cieplnej lub mapy fiksacji. Mapy cieplne (ang. *heatmaps*) przedstawiają obraz widziany przez osobę badaną, na który naniesione są obszary różnej wielkości i intensywności koloru. Obszary te znajdują się w miejscach, na które patrzyła osoba badana. Wielkość oraz intensywność naniesionych obszarów zależy od częstotliwości i czasu skupienia wzroku na oglądanym elemencie. *Gorące obszary*, to te które w największym stopniu zainteresowały badanego [6, s.123]. Przykład mapy cieplnej przedstawia rys. 2.10. Wynika z niej, że znacznie większe



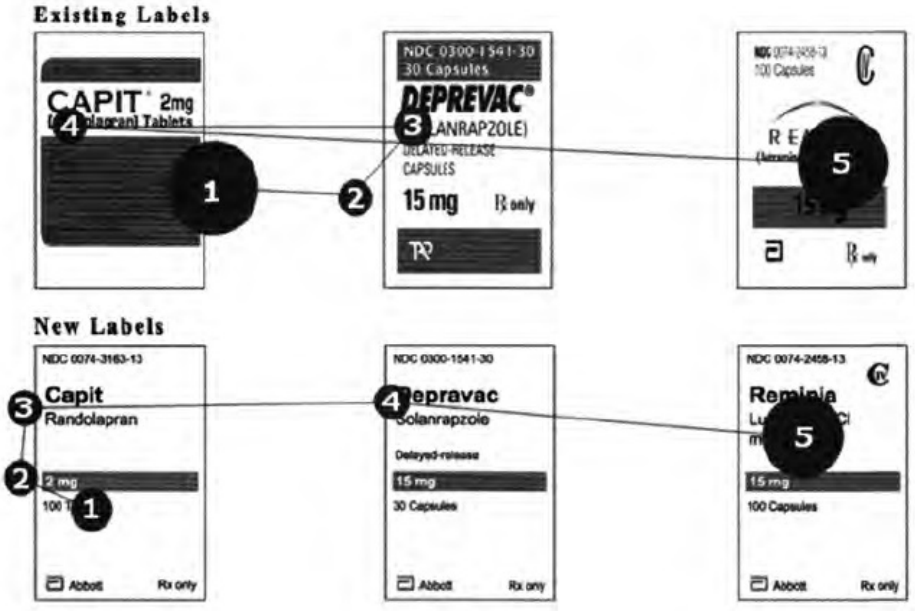
Rys. 2.10 Mapa cieplna uzyskana podczas badań serwisu Google. Źródło: [12].

zainteresowanie, a więc także liczba fiksacji oka, skupiona była w lewej górnej części wyników wyszukiwania a ignorowane są linki sponsorów umieszczonych w prawej kolumnie [4]. Z tego powodu Google zmienił sposób wyświetlania reklam, które zostały przeniesione na lewą stronę, nad wynik wyszukiwań. Oczywiście konsekwencją tego może być utrata zaufania klientów.

Często mapy, które zawierają jedynie gorące obszary są nazywa się mapami uwagi lub zainteresowania. Widoczne są tylko te obszary, które zainteresowały badanego - pozostała część jest przesłonięta. Przykładem mapy uwagi jest rys. 2.11. Mapa ta pochodzi również z badań serwisu Google.

Rys. 2.11 Mapa uwagi uzyskana podczas badań serwisu Google. Źródło: [13].

Innym rodzajem map są mapy fiksacji. Podobnie jak mapy cieplne, w tle zamieszany jest obraz oglądany przez badanego. Na obraz ten zostają naniesione linie oraz koła. Linie ukazują drogę przemieszczania się wzroku w trakcie sakkad, natomiast średnice okręgów są proporcjonalne do czasu trwania fiksacji [6, s.123]. Często również dodaje się liczby w okręgach, które wskazują kolejność fiksacji. Przykład mapy fiksacji przedstawiony na rys. 2.12 ukazuje wynik badań przeprowadzonych z powodu wprowadzenia nowej etykiety leku, wraz z analizą porównawczą zmiany zainteresowania jej poszczególnymi elementami w odniesieniu do poprzedniej etykiety. W tym przypadku zarejestrowany ruch oka pomógł potwierdzić zwiększoną wydajność wizualnego poszukiwania informacji o leku na nowej etykiecie. Wpływ miało lepsze pozycjonowanie tekstu, jaśniejsze tło i użycie bardziej spój-



Rys. 2.12 Przykład mapy fiksacji z badań nad wprowadzeniem nowej etykiety leków. Źródło: [4, s.273].

nych czcionek [4, s. 274].

Omawiając temat użycia okulografii w różnych aspektach życia, należy pamiętać o bardzo szerokim zastosowaniu w ośrodkach naukowych i centach badawczych. W zdecydowanej większości badania te dotyczą takich dziedzin nauki, jak: psychologia, neurologia, kognitywistyka, socjologia ale również informatyka i automatyka. Zdarza się także, że okulograf wykorzystuje się, jako urządzenie pomocnicze lub dodatkowe w badaniu. Czasem jest on łączony np. z EEG[6]. Takie połączenie znacznie poszerza możliwości badawcze takich systemów, co sprawia, że nie ma jednoznacznie określonych barier zastosowania danego sprzętu. Przykłady prowadzonych badań z użyciem eyetrackingu oraz systemu GCAF zostały przedstawione w ostatnim rozdziale pracy.

Eyetracking dostarcza również nowych możliwości w wielu dziedzinach psychologii. W psychologii kognitywnej i kognitywistyce prowadzone są badania w zakresie percepcji bodźców wzrokowych, sposobie postrzegania informacji w zależności od jej formy oraz interakcji człowieka z komputerem a także omówione wcześniej badania, dotyczące uwagi kierowców. Psychologia rozwojowa zajmuje się badaniem rozwoju koordynacji wzrokowo-ruchowej, alokacji uwagi oraz zależności pomiędzy sposobem sterowania ruchami a rozu-

[6]EEG, czyli elektroencefalografia, która polega na nieinwazyjnym badaniu bioelektrycznej czynności mózgu

mieniem tekstu. Przy użyciu eyetrackingu bada się również autyzm. Psychologia eksperymentalna wykorzystuje eyetracking w zakresie rozpoznawania twarzy, percepcji wzrokowej obiektów oraz różnic korzystania z funkcji wzrokowo-przestrzennych, zarówno przez osoby zdrowe, jak i osoby z uszkodzonym układem nerwowym. Psycholingwistyka prowadzi badania w zakresie trudności czytania tekstu. Tworzy się programy treningowe, których celem jest wspomaganie umiejętności czytania. Neuropsychologia korzysta z eyetrackerów do analizy strategii oglądania scen obrazów zarówno przez osoby chore, jak i osoby zdrowe. Ta dziedzina psychologii zajmuje się także wykorzystaniem eyetrackingu wraz z EEG. Eyetracking w psychologii wykazuje się dużymi możliwościami zastosowania, które są ciągle rozwijane [16].

3. Język GIML

Większość oprogramowania dostępnego na rynku jest dostarczana w postaci pakietu zawierającego skompilowany program oraz instrukcję obsługi. Kod źródłowy pozostaje własnością firmy tworzącej oprogramowanie. To oznacza, że klient otrzymuje program, który posiada ustalone funkcjonalności. Zmiany lub rozbudowa programu wymaga złożenia nowego zamówienia w tej samej firmie. Dzięki takim zabiegom firmy wytwarzające oprogramowanie wiążą ze sobą swoich klientów. Rozdział ten jest poświęcony technologii, która zmienia sposób podejścia do wytwarzania oprogramowania, dając większe możliwości końcowemu użytkownikowi oprogramowania.

3.1 Nowe podejście do tworzenia aplikacji

Tworzenie oprogramowania przy użyciu nowych narzędzi developerskich jest nastawione zarówno na wydajność, jak i uniwersalność. Uniwersalność ta daje przenaszalność kodu źródłowego pomiędzy różne środowiska uruchomieniowe. Dzięki temu program napisany na urządzenie desktopowe, mógł także działać na urządzeniach mobilnych. Z takim udogodnieniem wiążą się różne problemy dotyczące zgodności, standaryzacji, obsługi bibliotek, itp. Ważnym czynnikiem tej uniwersalności są języki deklaratywne, które w odróżnieniu od języków imperatywnych nie wymagają instrukcji, co program ma po kolei wykonać. Określa się w nich jedynie to, co chce się uzyskać [17]. Języki deklaratywne są coraz powszechniejsze ze względu na swoją prostotę użycia. W odróżnieniu od języków imperatywnych, programista nie musi wiedzieć, jakie instrukcje wykonać po kolei, aby osiągnąć zamierzony cel. Przykładem takiego języka jest XAML stworzony przez firmę Microsoft, który służy do opisu interfejsu (wyglądu okna aplikacji). Jest to deklaratywny język opisu, który poprzez pliki z kodem źródłowym oddziela UI[1] od logiki, czyli aplikacji [14]. Obecnie wiele środowisk wspiera XAML, dzięki czemu poznając jeden język można tworzyć wizualną stronę różnych typów aplikacji. Problem tkwi w tym, że wciąż jest to język, który podlega kompilacji przez wspólne środowisko uruchomieniowe na plat-

[1]UI (user interface), czyli interfejs użytkownika, który dotyczy tej sekcji programu, która odpowiada za bezpośrednią interakcję z użytkownikiem.

formie .NET, czyli przez CLR[2]. Nadal więc pozostaje problem ograniczonych możliwości użytkownika w zakresie rozbudowy programu.

Język GIML, którego pełna nazwa to Gaze Interaction Markup Language, jest również opisem interfejsu aplikacji, podobnie jak XAML oparty na XML, ale przedstawia zupełnie nowy sposób podejścia do tworzenia wizualnej strony programów. Twórcą języka jest dr hab. Jacek Matulewski z Uniwersytetu Mikołaja Kopernika w Toruniu, który po raz pierwszy zaimplementował jego obsługę w platformie GCAF. Głównym założeniem podczas projektowania tego deklaratywnego języka była możliwość tworzenia własnego wyglądu aplikacji przez jego końcowych użytkowników.

3.2 Struktura GIML

Struktura języka GIML oparta jest na znacznikach znanych z XML[3]. GIML jest więc językiem znaczników. To duża zaleta, ponieważ pliki XML są bardzo popularnym formatem zapisu danych oraz wymiany informacji pomiędzy różnymi środowiskami programistycznymi. Tworzenie plików z użyciem GIMLa jest więc prostym tworzeniem plików z rozszerzeniem XML, które zawierają odpowiednią strukturę znaczników określonych standardem języka GIML. Można do tego użyć jednego z edytorów XML, które ułatwiają stworzenie odpowiedniej struktury zgodnie ze standardem. Kolejną zaletą takiego podejścia jest fakt, że stworzony za pomocą GIML opis interfejsu aplikacji nie jest kompilowany na etapie budowania programu. Pliki te wczytuje się dopiero po uruchomieniu platformy GCAF, która jest interpreterem języka GIML. Ładowanie plików w czasie trwania programu jest możliwe dzięki parserowi GIML, który odczytuje plik XML, przetwarza strukturę znaczników zaczynając od tych, które mają najwyższą pozycję w hierarchii, a następnie tworzy obiekty z parametrami określonymi przez atrybuty znaczników. Obiekty podlegają późniejszej analizie i na ich podstawie tworzony jest obraz wyświetlany użytkownikowi. Jedną z funkcji dostępnych w parserze języka jest walidacja wprowadzanych danych, czyli sprawdzanie poprawności wprowadzanych znaczników i ich atrybutów. Jest to narzędzie wykorzystywane podczas tworzenia własnych plików GIML. Działanie parsera

[2]CLR (Common Language Runtime) kompiluje kod pośredni na postać zrozumiałą dla danego typu procesora i następnie go wykonuje [7, s.137]

[3]XML (Extensible Markup Language) to uniwersalny język znaczników uporządkowanych w logiczne struktury, które reprezentują różne dane

GIML można porównać do powszechnie stosowanej deserializacji obiektów. Polega ona na odczycie z pliku (np. XML, JSON) struktury danych a następnie "przetłumaczenie" ich na dany obiekt w wybranym języku programowania. Takie zabiegi często wykorzystuje się w aplikacjach webowych, gdzie obiekty z wypełnionymi polami są serializowane (proces odwrotny do deserializacji) do wspólnego formatu plików, a następnie przesyłane łączem sieciowym, aby po drugiej stronie można było ten plik odczytać i zdeserializować, co prowadzi do uzyskania pierwotnego obiektu z danymi. Ważne jest, że obiekty, które powstają po obu stronach łącza nie muszą być tworzone w tym samym języku programowania. Cały proces tłumaczenia struktury obiektów zależy od użytych serializatorów. Każdy ze współczesnych języków programowania wysokiego poziomu udostępnia wbudowane biblioteki z serializatorami, z obsługą różnego typu plików. Nie ma również przeszkód, aby stworzyć własny serializator do danych. W przypadku GIMLa odpowiednikiem serializatora jest osoba, która tworzy strukturę w języku GIML i zapisuje ją do pliku XML.

Prostota języka kryje się w jego strukturze, która składa się z zaledwie kilku znaczników ułożonych w logicznej hierarchii. Główną funkcjonalność, a zarazem i różnorodność interfejsu aplikacji otrzymuje się poprzez dodanie do znaczników odpowiednich atrybutów (parametrów, własności). Opisują one nazwę, rozmiar i inne statyczne własności obiektów na ekranie. Wyróżnikiem GIML jest to, że atrybuty mogą również określać zachowanie obiektów na ekranie oraz reakcje na zachowania użytkownika, w tym przede wszystkim na spojrzenie. Stworzenie okna aplikacji wymaga od użytkownika opisania za pomocą znaczników GIML tego, co i gdzie ma się pojawić na ekranie oraz jaką dany obiekt ma pełnić funkcję. Jeśli zajdzie potrzeba dołożenia lub całkowitej zmiany aplikacji, wystarczy odpowiednio zmodyfikować plik GIML i wczytać go ponownie w platformie GCAF. Sposób ten rozwiązuje dość powszechny problem występujący w komunikacji pomiędzy osobami tworzącymi oprogramowanie a klientem, którego wizja funkcjonowania i wyglądu programu nie zawsze pokrywa się z tym, o czym myślą jego twórcy. Wpływa on także na zmniejszenie kosztów. Ważną cechą języka GIML jest jego wsparcie dla wielu języków.

Plik GIML, który jest plikiem XML na początku powinien zawierać deklarację XML wraz z atrybutem `version` (określa wersję języka - 1.0 lub 1.1) oraz opcjonalnymi atrybutami - `encoding` (określa sposób kodowania) oraz `standalone` (określa typ dokumentu). Ustawienie drugiego parametru na "yes" oznacza, że dokument nie zawiera innych plików, które musiałyby zostać dołączone przy przetwarzaniu danych) [15]. Przykład takiej

deklaracji:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Tworzenie struktury w języku GIML rozpoczyna się od znacznika[4] `ustawienia`, który jest nadrzędnym elementem w logicznej strukturze pliku, czyli tzw. *rootem* w strukturze XML. Mając zdefiniowany znacznik `ustawienia` kolejnymi podrzędnymi znacznikami w pliku są `rysunki`, `dźwięki`, `filmy`, czyli zasoby oraz `sceny`, czyli opis okna. Każdy z

```
<?xml version="1.0" encoding="UTF-8"?>
<ustawienia [atrybuty]>
    <filmy>
        <film [atrybuty] />
        <film [atrybuty] />
    </filmy>
    <rysunki>
        <rysunek [atrybuty] />
    </rysunki>
    <dźwięki>
        <dźwięk [atrybuty] />
        <dźwięk [atrybuty] />
    </dźwięki>
    <sceny [atrybuty]>
        <scena [atrybuty]>
            <obszar [atrybuty]>
                <reakcja [atrybuty] />
            </obszar>
            <obszar [atrybuty]>
                <aktywacja [atrybuty] />
                <reakcja [atrybuty] />
            </obszar>
        </scena>
        <scena [atrybuty]>
            <obszar [atrybuty]>
                <aktywacja [atrybuty] />
                <reakcja [atrybuty] />
            </obszar>
            <obszar [atrybuty]>
                <aktywacja [atrybuty] />
                <reakcja [atrybuty] />
            </obszar>
        </scena>
    </sceny>
</ustawienia>
```

Rys. 3.1 Schemat struktury plików XML z użyciem GIML

tych elementów zawiera odpowiednio znaczniki `rysunek`, `dźwięk`, `film` oraz `scena`. Trzy pierwsze elementy są ostatnimi w hierarchii, natomiast element `scena` może zawierać elementy o nazwie `obszar`. Każdy z obszarów może również zawierać znacznik `aktywacja` oraz `reakcja`. Zarys struktury najlepiej przedstawia schemat pliku widoczny na rys. 3.1.

[4]Każdy element języka XML zaczyna się i kończy znacznikiem. Deklaracja, która występuje pomiędzy znacznikiem otwierającym i zamykającym to zawartość elementu

3.3 Znacznik `ustawienia`

Zgodnie w wcześniejszym opisem ogólnej struktury GIML, element `ustawienia` jest elemen1tem położonym najwyżej w hierarchii pliku. Jak sama nazwa wskazuje element ten jest odpowiedzialny za ustawienia ekranu wyświetlania. Ustawienia te są definiowane przy pomocy czterech atrybutów (tab. 3.1). Pierwszym z nich jest `katalog`, który określa bezwzględną ścieżkę do katalogu zawierającego zewnętrzne zasoby, które są wykorzystywane przez inne elementy w pliku GIML. Ważne jest, aby wprowadzona ścieżka była poprawna, ponieważ na etapie parsowania pliku GIML, parser zwróci błąd o braku podanych plików na dysku.

Drugi atrybut `biblioteka` służy do ustawienia bezwględnej ścieżki do pliku biblioteki, która obsługuje zdarzenia zewnętrzne. Parametr ten jest istotny w przypadku, kiedy użytkownik chce rozszerzyć możliwości GIML za pomocą biblioteki .NET. Podając atrybut `biblioteka`, konieczne jest również dodanie trzeciego atrybutu ustawień mianowicie elementu `klasa`. Zawiera on nazwę klasy, która definiuje metody zdarzeniowe. Opis zdarzeń znajduje się w dalszej części pracy.

Ostatnim z atrybutów elementów `ustawienia` jest `język`, którego wartością jest kod języka (np. "pl", "en"). Parametr ten jest ważny jeśli użytkownik przygotuje plik GIML w innym języku niż polski. Wszystkie znaczniki, atrybuty, itd. mają w wielu językach swoje tłumaczenia. Struktura GIML się jednak nie zmienia, dlatego nie ma problemu z ich weryfikacją. Przykładowo znacznik `obszar` ma w języku angielskim odpowiednik `area`.

Wszystkie z wymienionych atrybutów są opcjonalne. Najczęściej jednak dodaje się atrybut `katalog`, gdyż większość plików GIML wykorzystuje zewnętrzne zasoby. W przypadku atrybutu `język` domyślnym jest język polski. Przykładowa implementacja elementu `ustawienia` wygląda następująco:

```
<ustawienia katalog="t:\Zasoby\" język="pl"
    biblioteka="..\..\..\BibliotekaZdarzeń.dll" klasa="PrzestrzeńNazw.Klasa">
```

Tablica 3.1 Znacznik `ustawienia` i jego atrybuty

Atrybut [pl]	**Atrybut [en]**	**Opis**	**Domyślna wartość**
`katalog`	`folder`	Bezwzględna ścieżka do katalogu z zasobami	Katalog pliku wykonywalnego GCAF
`biblioteka`	`library`	Bezwzględna ścieżka do pliku biblioteki obsługującej zdarzenia zewnętrzne	Katalog pliku wykonywalnego GCAF
`klasa`	`class`	Nazwa klasy zawierająca definicję metod zdarzeniowych	*brak*
`język`	`language`	Zawiera kod używanego języka	pl

3.4 Znaczniki zasobów

Zasoby (ang. *resources*) są elementami GIML reprezentującymi zewnętrzne obiekty, do których znajdują się odwołania w programie. Przykładowo chcąc na ekranie wyświetlić zdjęcie, wystarczy dodać je do zasobów, a następnie stworzyć obiekt, który będzie z tego zasobu korzystał. W języku GIML do definiowania zasobów służą znaczniki `film`, `rysunek` oraz `dźwięk`, które są zawarte w elementach odpowiednio wyższego poziomu, tzn. `filmy`, `rysunki` i `dźwięki`. Trzy ostatnie elementy nie posiadają żadnych atrybutów. Dodanie do nich atrybutów spowoduje zwrócenie błędu przez parsera.

Znacznik `rysunek` pozwala na odwołanie do różnego rodzaju obrazków, zdjęć oraz animacji. Jego właściwości są określane przy użyciu siedmiu atrybutów. Pierwszy z nich to `nazwa`. Jest łańcuchem znaków, który jednoznacznie identyfikuje dany zasób. Odwołanie w obszarze do danego zasobu odbywa się właśnie poprzez jego nazwę. Drugim parametrem jest `ścieżka`, która zawiera względną ścieżkę do pliku z zasobem, który ma zostać użyty. Ścieżka absolutna jest tworzona przez powiązanie z omówionym wcześniej atrybutem `katalog` w elemencie `ustawienia`. Przykładowo, jeśli wybrany plik obrazu o nazwie "obraz.jpg" znajduje się w katalogu "Zasoby" na dysku C, to zaleca się, aby wartość atrybutu `katalog` równa była "C:\\Zasoby", a atrybutu `ścieżka` "obraz.jpg". Atrybut `przechowujWPamięci` pozwala na załadowanie elementu `rysunek` do pamięci operacyjnej

aby przy kolejnych odwołaniach skrócić czas dostępu do tego zasobu. Kolejny atrybut to `kluczPrzezroczystości`. Określa kolor, który zostanie wyświetlony, gdy obiekt będzie zawierał elementy przezroczyste. Pozostałe trzy atrybuty odpowiadają za ustalenie parametrów animacji oraz filmów. Pierwszy z nich `okresOdtwarzania` określa, przez jaki czas animacja ma być odtwarzana. Pozostałe dwa atrybuty, czyli `odtwarzanieOdRamki` i `odtwarzanieDoRamki` pozwalają odtwarzać wybrany fragment animacji. Przykład użycia elementu `rysunek`:

```
<rysunki>
    <rysunek nazwa="elmo" ścieżka="rysunki\elmo.png" kluczPrzezroczystości="Fuchsia"
             okresOdtwarzania="1200" odtwarzanieOdRamki="50" odtwarzanieDoRamki="75"
             przechowujWPamięci="tak" />
</rysunki>
```

Drugim rodzajem zasobów są dźwięki, które są opisywane przez znacznik `dźwięk`. Podobnie, jak w przypadku znacznika `rysunek`, element dźwięku również posiada atrybuty `nazwa` i `ścieżka`. Różni się jednak pozostałymi atrybutami, które odnoszą się bezpośrednio do sposobu odtwarzania dźwięku. Atrybut `wTle` przyjmuje wartość "tak" lub "nie" określając w ten sposób, czy ścieżka dźwiękowa ma służyć, jako podkład dźwiękowy, czy też ma to być dźwięk wydobywający się tylko przy zajściu określonych zdarzeń. Drugim parametrem jest `głośność`, która odpowiada za poziom emisji dźwięku. Przyjmowana wartość jest z zakresu od 0.0 do 1.0. Ostatni atrybut to `liczbaPowtórzeń`, którego wartość decyduje o tym, ile razy dźwięk ma być odtwarzany (w pętli). Ustawienie wartości tego atrybutu na 0 oznacza odtwarzanie ciągłe. Użycie znacznika `dźwięk`:

```
<dźwięki>
    <dźwięk nazwa="chimes" ścieżka="dźwięki\chimes.wav"
            liczbaPowtórzeń="3" wTle="nie" głośność="0.1" />
</dźwięki>
```

Ostatnim z zasobów są filmy opisywane przez znacznik `film`. Odwołują się one do plików filmowych w formacie AVI. Atrybuty, które posiadają występują również w przypadku dźwięków. Są to `nazwa`, `ścieżka`, `liczbaPowtórzeń`, `wTle` oraz `głośność`. Przykład implementacji elementu `film`:

```
<filmy>
    <film nazwa="farscape" ścieżka="filmy\farscape.avi" kluczPrzezroczystości="Fuchsia"
          liczbaPowtórzeń="2" głośność="1" wTle="tak"/>
</filmy>
```

Tablica 3.2 Znacznik `rysunek` i jego atrybuty

Atrybut [pl]	**Atrybut [en]**	**Opis**	**Domyślna wartość**
`nazwa`	`name`	Łańcuch znaków jednoznacznie identyfikujący dany zasób	`brak`
`ścieżka`	`path`	Względna ścieżka do pliku zasobu, który ma zostać użyty	Katalog pliku wykonywalnego GCAF
`przechowujWPamięci`	`keepInMemory`	Przetrzymanie obiektu w pamięci operacyjnej do późniejszych odwołań w celu zwiększenia szybkości ładowania zasobu	nie
`kluczPrzezroczystości`	`transparencyKey`	Kolor podłoża wyświetlanego, jeśli obiekt zawiera obszary przezroczyste	Fuchsia
`okresOdtwarzania`	`runningPeriod`	Czas odtwarzania animacji w ms	1000
`odtwarzanieOdRamki`	`runFromFrame`	Wybór numeru klatki, od której ma się rozpocząć odtwarzanie animacji	0
`odtwarzanieDoRamki`	`runToFrame`	Wybór numeru klatki, do której animacja ma się odtwarzać	-1

Tablica 3.3 Znaczniki `dźwięk` oraz `film` i ich atrybuty

Atrybut [pl]	**Atrybut [en]**	**Opis**	**Domyślna wartość**
`nazwa`	`name`	Łańcuch znaków jednoznacznie identyfikujący dany zasób	`brak`
`ścieżka`	`path`	Względna ścieżka do pliku zasobu	Katalog pliku wykonywalnego GCAF
`wTle`	`inBackground`	Ustawienie ścieżki dźwiękowej jako podkład dźwiękowy	nie
`głośność`	`volume`	Poziom emisji dźwięku	1.0
`liczbaPowtórzeń`	`repetitionNumber`	Ile razy dźwięk ma być odtwarzany	1

3.5 Znaczniki `sceny` i `scena`

Kolejne znaczniki GIML odpowiadają za to, co zostanie wyświetlone użytkownikowi na ekranie. Nadrzędnym elementem jest element `sceny`, który powinien zawierać znaczniki `scena`.

Znacznik `sceny` zawiera wszystkie elementy `scena`, które są zawarte w danym pliku GIML. Podsiada cztery atrybuty, które decydują o poprawności wyświetlanego obrazu. Pierwszy to `nazwaScenyDomyślnej`. Zawiera odwołanie do jednego z pod-elementów typu `scena` (poprzez jego nazwę), która ma zostać pokazana tuż po uruchomieniu aplikacji. Podobnym atrybutem jest `nazwaScenyPauzy`, który wskazuje scenę wyświetlaną użytkownikowi, kiedy zostanie naciśnięty klawisz pauzy (RR). Pozostałe dwa parametry są ściśle ze sobą powiązane i określają szerokość oraz wysokość oryginalnego ekranu - ich nazwy to odpowiednio `oryginalnyRozmiarEkranuX` i `oryginalnyRozmiarEkranuY`. Prawidłowe ustawienie tych parametrów, powinno uwzględniać rozdzielczość ekranu, na którym będzie wyświetlana aplikacja, co jest ważne ze względu na użycie eyetrackera. Brak zachowania proporcji ekranu, a także używanie niestandardowych wymiarów, może po-

wodować zniekształcenie obrazu. W przypadku, kiedy używa się monitorów o różnych rozdzielczościach lub rozdzielczość monitora jest inna niż standardowa rozdzielczość eyetrackera, powstają błędy przy analizie pozycji wzroku. Błędy te polegają na przesunięciu punktu wzroku w odniesieniu do położenia rzeczywistego. Ustawienie parametru `oryginalnyRozmiarEkranuX/Y` pozwalają uniknąć tego uniknąć. Stosowanie samego powiększenia okna aplikacji do wymiarów monitora bez wcześniejszego zdefiniowania tych parametrów, również będzie obarczone błędem pomiaru pozycji wzroku. Najlepsze rezultaty pomiarów uzyskuje się, gdy rozdzielczość eyetrackera oraz monitora są takie same.

Element `scena` jest elementem podrzędnym względem znacznika `sceny`. Posiada dziesięć atrybutów. Jednym z nich jest atrybut `nazwa`, który podobnie, jak w przypadku zasobów pozwala na odwołania do danej sceny. Kolejne cztery parametry dotyczą wyglądu sceny. Pierwszy z nich to `kolorTła`, który określa kolor tła sceny. Parametr ten może być nieistotny, jeśli ustawiony zostanie drugi z atrybutów - `nazwaRysunkuTła`, zawierający nazwę rysunku z zasobów. Następny parametr to `kolorZaczernienia`, którego wartość określa kolor używany do przyciemnienia ekranu. Jego nieprzezroczystość jest określana parametrem `stopieńZaczernienia`. Każda ze scen może mieć określony dźwięk tła, który jest ustawiany za pomocą atrybutu `nazwaDźwiękuTła`, stanowiącego odwołanie do jednego z zasobów dźwięku. Ostatnia z własności sceny dotyczy zdarzenia zewnętrznego, wywoływanego przy wychodzeniu ze sceny, które może być aktywowane przez znacznik `poZmianieSceny` - zawiera nazwę metody z załadowanej biblioteki zdarzeń. Dodatkowymi atrybutami odpowiedzialnymi za obszary są `blokowanieObszarówPrzyZaczernieniu`, który pozwala zablokować wszystkie zaczernione obszary na scenie oraz `nazwaObszaruWłączanegoPoWyłączeniuWszystkichObszarów` pozwalający określić ostatni aktywny obszar na scenie, który najczęściej jest używany, jako przełącznik do następnej sceny. Ostatnim atrybutem jest `scenaWzorcowa`, która pozwala na dziedziczenie elementu `scena`.

Odpowiednie przygotowanie scen oraz przejść między nimi pozwala tworzyć ciekawe animacje interaktywne sterowane wzrokiem. Przykładem są interaktywne gry dla dzieci, które pozwalają poznawać dziecku świat przez podążanie wzrokiem za pojawiającymi się obiektami.

Poglądowa implementacja elementu `scena` w elemencie `sceny`:

```
<sceny nazwaScenyDomyślnej="0" nazwaScenyPauzy="2" oryginalnyRozmiarEkranuX="1024"
       oryginalnyRozmiarEkranuY="768">
    <scena nazwa="0" kolorTła="Lime" nazwaRysunkuTła="tło" nazwaDźwiękuTła="tło"
           scenaWzorcowa="szablon" stopieńZaczernienia="200" kolorZaczernienia="Black"
           blokowanieObszarowPrzyZaczernieniu="tak|nie"
           nazwaObszaruWłączanegoPoWyłączeniuWszystkichObszarów="bramaDoNastępnejSceny"
           poZmianieSceny="scena0_poZmianieSceny">
        <obszar
        (zob. poniżej)
        />
    </scena>
</sceny>
```

Tablica 3.4 Znacznik `sceny` i jego atrybuty

Atrybut [pl]	**Atrybut [en]**	**Opis**	**Domyślna wartość**
`nazwaScenyDomyślnej`	`nameOfDefaultScene`	Odwołanie do elementu `scena`, która jest wyświetlana tuż po uruchomieniu aplikacji	Pierwsza zdefiniowana scena w pliku GIML
`nazwaScenyPauzy`	`nameOfPauseScene`	Odwołanie do elementu `scena`, która jest wyświetlana po włączeniu pauzy	*brak*
`oryginalnyRozmiarEkranuX`	`originalScreenSizeX`	Szerokość oryginalnego ekranu	Szerokość monitora z eksperymentem
`oryginalnyRozmiarEkranuY`	`originalScreenSizeY`	Wysokość oryginalnego ekranu	Wysokość monitora z eksperymentem

Tablica 3.5 Znacznik `scena` i jego atrybuty

Atrybut [pl]	**Atrybut [en]**	**Opis**	**Domyślna wartość**
`nazwa`	`name`	Łańcuch znaków jednoznacznie identyfikujący daną scenę	`brak`
`kolorTła`	`backgroundColor`	Kolor tła sceny	Biały
`nazwaRysunkuTła`	`nameOfBackground Image`	Ustawienie rysunku jako tło sceny	`brak`
`kolorZaczernienia`	`blackoutColor`	Kolor używany do przyciemnienia ekranu	czarny
`stopieńZaczernienia`	`blackoutDegree`	Poziom nieprzezroczystości zaczernienia ekranu	0
`nazwaDźwiękuTła`	`nameOfBackground Sound`	Dźwięk odtwarzany w tle	`brak`
`poZmianieSceny`	`onSceneChanged`	Nazwa metody zdarzeniowej wywoływana po zmianie sceny	`brak`
`blokowanieObszarów PrzyZaczernieniu`	`blockRegions DuringBlackout`	Zablokowanie obszarów na scenie, które są przyciemnione	nie
`nazwaObszaru WłączanegoPo Wyłączeniu WszystkichObszarów`	`nameOfRegion EnabledAfterAll RegionsAreDisabled`	Obszar aktywowany po wyłączeniu wszystkich obszarów na scenie	`brak`
`scenaWzorcowa`	`templateScene`	Dziedziczona scena	`brak`

3.6 Znacznik `obszar`

Elementem GIML opisującym obiekty wyświetlane na scenie jest znacznik `obszar`. Definicja obszaru określa go, jako obiekt znajdujący się na danej scenie. Jest więc elementem podrzędnym elementy `scena`. Obszary posiadają własność dziedziczenia, dzięki czemu można raz zaimplementować dany obszar i używać go w kliku miejscach pliku GIML, bez potrzeby powtarzania jego całej definicji.

Obszary są najbardziej rozbudowanymi elementami występującymi w języku GIML. Posiadają wiele atrybutów, które opisują ich własności. Podstawowym jest oczywiście atrybut `nazwa`. Obszar jest domyślnie włączony, ale można to ustawić. Przykładowo na scenie może być obiekt, który jest aktywowany dopiero po zajściu jakiegoś zdarzenia. Atrybutem, który za to odpowiada jest `włączony`. Ściśle powiązane z aktywnością obszaru są parametry `opóźnienieWłączenia` i `opóźnienieWyłączenia`, które pozwalają aktywować / dezaktywować obszar po upływie określonego czasu od wystąpienia zdarzenia.

Istotnym atrybutem obiektów jest określenie ich kształtu, położenia na ekranie oraz rozmiaru. Właściwości te ustalane są odpowiednio przez: `kształt`, `położenieŚrodkaX`, `położenieŚrodkaY`, `rozmiarX` i `rozmiarY`. Domyślnymi kształtami są proste figury geometryczne, które wyznaczają zewnętrzną granicę obszarów: prostokąt, koło, elipsa. W przypadku, gdy użytkownik chce wyświetlić rysunek, który ma skomplikowany kształt to wybiera na kształt prostokąt, w który zostanie wpisany wybrany rysunek. Należy zawsze dopasować wymiary obszaru do obiektu, który ma on reprezentować. W przypadku użycia GIML, w aplikacjach sterowanych wzrokiem należy uwzględnić dokładność użytego eyetrackera, by uniknąć sytuacji zbyt małych lub zbyt blisko siebie położonych obszarów, co prowadzi do błędów odczytu położenia wzroku na niewłaściwym obszarze.

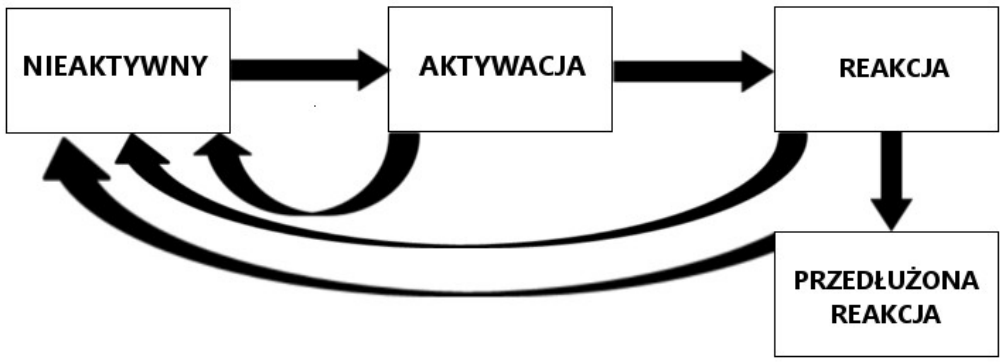
Zasoby (dźwięki, rysunki i filmy) mogą być przypisane do obszarów. Służą do tego atrybuty pozwalające wskazać odpowiedni zasób. Najważniejszym z nich są `nazwaRysunku` i `nazwaDźwięku` pozwalające na wskazaniu rysunku lub dźwięku, który ma zostać użyty. W przypadku rysunków można podać rozmiar i położenie względem obszaru poprzez: `rozmiarRysunkuX`, `rozmiarRysunkuY`, `przesunięcieŚrodkaRysunkuX` oraz `przesunięcieŚrodkaRysunkuY`.

Obszary posiadają także atrybuty, które kontrolują ich dynamikę. Są to: `typAkcji` określający zdarzenie, gdy punkt wzroku znajdzie się na obszarze (np. przejście do nowej

sceny), `grubośćRamki` oraz `kolorRamki`, która może pojawić się np. aktywacji obszaru wzrokiem; `animacjaObszaru` oraz `animacjaRysunku`, które za pomocą zmiennych "tak" lub "nie" umożliwiają stosowanie animacji; `okresAnimacji` i `amplitudaAnimacji` charakteryzują parametry odtwarzania ramek animacji; `wyłączPoZakończeniu` pozwala na dezaktywację obszaru po wystąpieniu określonego zdarzenia; `automatycznaReakcjaPoCzasie` służy wymuszeniu wywołania zdarzenia reakcji po czasie (zdarzenia reakcji oraz aktywacji zostaną omówione w dalszej części pracy); `nazwaObszaruWłączanegoPoRozpoczęciu`, `nazwaObszaruWyłączanegoPoZakończeniu` i `nazwaObszaruWłączanegoPoZakończeniu` umożliwiają z poziomu obszaru aktywację innych obszarów znajdujących się na scenie. Atrybut `nazwaDocelowejSceny` jest ważny, bo powoduje przejście pomiędzy różnymi scenami.

Pozostają również pod-elementy związane z domyślnymi zdarzeniami, tzn. aktywacją i reakcją. Aktywacja to stan, gdy punkt wzroku wejdzie na pole danego obszaru. Taka sytuacja występuje, np. podczas analizy tego co znajduje się na scenie, gdy wzrok przemierza obecne na niej obiekty obiektach. Drugi typ zdarzenia, reakcja, odpowiada sytuacji, kiedy wzrok pozostanie na obszarze dłużej, niż pewien określony czas. Oznacza to, że każdy obszar najpierw podlega aktywacji i jeśli wzrok nie opuści obszaru przez okres od aktywacji do reakcji, to wywoływane jest zdarzenie reakcji. Sterowanie tymi zdarzeniami z poziomu obszaru jest możliwe za pomocą atrybutów: `warunekZakonczeniaReakcji`, `czasUtrzymywaniaReakcji`, który może wydłużyć reakcję nawet pomimo zmiany położenia wzroku. Atrybut `odłóżPrzejścieDoScenyDoZakończeniaReakcji` jest stosowany w obszarach, które pełnią funkcję przełączników pomiędzy kolejnymi scenami. Kolejne trzy atrybuty `poAktywacji`, `poRozpoczęciuReakcji` oraz `poZakończeniuReakcji` mogą być wykorzystywane do uruchomienia zdarzeń zewnętrznych przy wystąpieniu aktywacji lub reakcji. Atrybuty `poPowrocieDoStanuNormalnego` oraz `poZmianieStanu` pozwalają wywołać zdarzenia zewnętrzne przy zmianie stanów obszarów. Zależności pomiędzy kolejnymi stanami obszarów przedstawia rys. 3.2.

Na tym nie kończy się jednak dostępna funkcjonalność dla obszarów. Wewnątrz ich można zdefiniować dwa pod-elementy - `aktywacja` oraz `reakcja`. Ponieważ każde z nich dotyczy stanów oka posiadają one wspólne atrybuty, które częściowo pokrywają się z tymi, które dostępne są dla samego elementu `obszar`. Różnica jednak polega na tym, że określone działania mogą zajść przy określonych okolicznościach lub mogą nie wystąpić



Rys. 3.2 Schemat zależności stanów aktywności obszarów.

wogóle. Uzależnia to wygląd i zachowanie sceny od zachowania użytkownika (jego spojrzenia). Znaczniki te mają pomóc w projektowaniu aplikacji przy sterowaniu wzrokiem. Są szczególnie ważne przy aplikacjach wykorzystywanych do badań reakcji pacjentów na dane bodźce wzrokowe. Atrybutami, które można wykorzystać w tych dwóch znacznikach są: `typAkcji`, `grubośćRamki`, `kolorRamki`, `nazwaRysunku`, `nazwaDźwięku`, `typAnimacji`, `okresAnimacji`, `amplitudaAnimacji`, `wyłączPoZakończeniu`, `nazwaObszaruWłączanegoPoRozpoczęciu`, `nazwaObszaruWłączanegoPoZakończeniu`, `nazwaObszaruWyłączanegoPoZakończeniu`, `nazwaDocelowejSceny`. Dodatkowymi atrybutami, które nie występują w przypadku obszarów są `znacznik`, `znacznikOpóźniony`, `opóźnienieZnacznikaOpóźnionego`, które są wykorzystywane do oznaczania zdarzeń wysyłanych do serwera eyetrackera (czyli do iViewX).

Tablica 3.6 Znacznik `obszar` i jego atrybuty

Atrybut [pl]	**Atrybut [en]**	**Opis**	**Domyślna wartość**
`nazwa`	`name`	Łańcuch znaków identyfikujący dany obszar	`brak`
`włączony`	`enabled`	Stan obszaru	Tak
`opóźnienieWłączenia`	`enablingRetardation`	Opóźnienie włączenia aktywności obszaru	0
`opóźnienieWyłączenia`	`opóźnienieWyłączenia`	Opóźnienie wyłączenia aktywności obszaru	-1
`kształt`	`shape`	Figura geometryczna wyznaczająca zewnętrzną granicę obszarów	`brak`
`położenieŚrodkaX`	`locationOfCenterX`	Współrzędna X środka obiektu	0
`położenieŚrodkaY`	`locationOfCenterY`	Współrzędna Y środka obiektu	0
`rozmiarX`	`sizeX`	Szerokość obiektu	0
`rozmiarY`	`sizeY`	Wysokość obiektu	0
`nazwaRysunku`	`nameOfImage`	Odwołanie do zasobu rysunek	`brak`
`nazwaDźwięku`	`nameOfSound`	Odwołanie do zasobu dźwięk	`brak`
`rozmiarRysunkuX`	`imageSizeX`	Szerokość rysunku	0
`rozmiarRysunkuY`	`imageSizeY`	Wysokość rysunku	0
`przesunięcie ŚrodkaRysunkuX`	`offsetOfImageCenterX`	Przesunięcie rysunku w poziomie	0

przesunięcie ŚrodkaRysunkuY	offsetOfImageCenterY	Przesunięcie rysunku w pionie	0
typAkcji	actionType	Zdarzenie, gdy punkt wzroku znajdzie się na obszarze	brak
grubośćRamki	borderWidth	Grubość ramki obszaru	0
kolorRamki	borderColor	Kolor ramki obszaru	czarny
animacjaObszaru	regionAnimationEnabled	Możliwość stosowania animacji obszaru	nie
animacjaRysunku	imageAnimationEnabled	Możliwość stosowania animacji rysunku	nie
okresAnimacji	animationPeriod	Okres animacji	-1
amplitudaAnimacji	animationAmplitude	Amplituda animacji	-1
wyłączPoZakończeniu	turnOffWhenFinished	Dezaktywacja obszaru po wystąpieniu danego zdarzenia	nie
automatyczna ReakcjaPoCzasie	automaticReaction AfterTime	Wymuszenie zdarzenia reakcji	-1
nazwaObszaru WłączanegoPo Rozpoczęciu	nameOfRegion EnabledWhen Started	Obszar włączany po rozpoczęciu danego zdarzenia	brak
nazwaObszaru WyłączanegoPo Zakończeniu	nameOfRegion DisabledWhen Finished	Obszar wyłączany po zakończeniu danego zdarzenia	brak
nazwaObszaru WłączanegoPo Zakończeniu	nameOfRegion EnableWhen Finished	Obszar włączany po zakończeniu danego zdarzenia	brak

nazwaDocelowej Sceny	nameOfTargetScene	Przejście do danej sceny	brak

Tablica 3.7 Znaczniki aktywacja oraz reakcja i ich atrybuty

Atrybut [pl]	**Atrybut [en]**	**Opis**	**Domyślna wartość**
typAkcji	actionType	Zdarzenie, gdy punkt wzroku znajdzie się na obszarze	brak
grubośćRamki	borderWidth	Grubość ramki obszaru	0
kolorRamki	borderColor	Kolor ramki obszaru	czarny
nazwaRysunku	nameOfImage	Odwołanie do zasobu rysunek	brak
nazwaDźwięku	nameOfSound	Odwołanie do zasobu dźwięk	brak
typAnimacji	animationType	Rodzaj animacji obszaru	brak
okresAnimacji	animationPeriod	Okres animacji	-1
amplitudaAnimacji	animationAmplitude	Amplituda animacji	-1
wyłączPoZakończeniu	turnOffWhenFinished	Dezaktywacja obszaru po wystąpieniu danego zdarzenia	nie
nazwaObszaru WłączanegoPo Rozpoczęciu	nameOfRegion EnabledWhen Started	Obszar włączany po rozpoczęciu danego zdarzenia	brak
nazwaObszaru WyłączanegoPo Zakończeniu	nameOfRegion DisabledWhen Finished	Obszar wyłączany po zakończeniu danego zdarzenia	brak

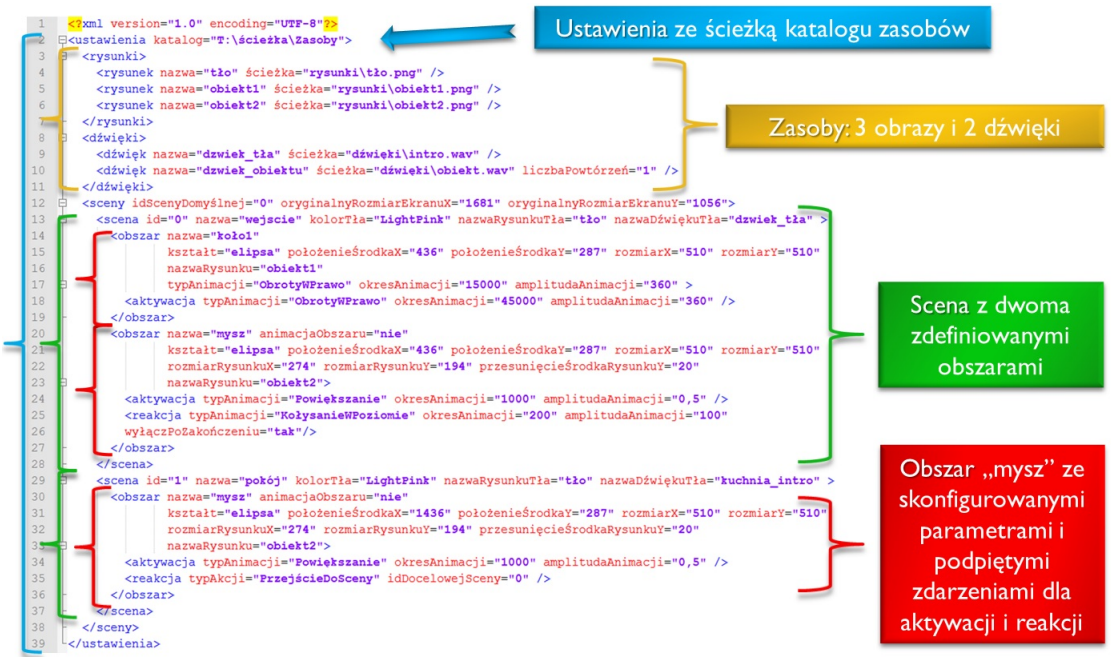
nazwaObszaru WłączanegoPo Zakończeniu	nameOfRegion EnableWhen Finished	Obszar włączany po zakończeniu danego zdarzenia	brak
nazwaDocelowej Sceny	nameOfTargetScene	Przejście do danej sceny	brak
znacznik	tag	Znacznik obszaru wysyłany do serwera ET	brak
znacznikOpóźniony	retardedTag	Znacznik obszaru wysyłany do serwera ET z opóźnieniem	brak
opóźnienieZnacznika Opóźnionego	retarderTag Retardation	Opóźnienie znacznik opóźnionego obszaru wysyłanego do serwera ET	brak

Przykład użycia elementu obszar oraz pod-elementów aktywacja i reakcja:

```
<obszar nazwa="pierwszy" włączony="tak|nie" animacjaObszaru="tak|nie" animacjaRysunku="tak|nie"
        możeAktywowaćZaczernienie="false"kształt="prostokąt|koło|elipsa" położenieŚrodkaX="200"
        położenieŚrodkaY="200" rozmiarX="200" rozmiarY="300" przesunięcieŚrodkaRysunkuX="10"
        przesunięcieŚrodkaRysunkuY="10" rozmiarRysunkuX="180" rozmiarRysunkuY="280"
        warunekZakończeniaReakcji=”WyjścieZObszaru|ZakończenieOdtwarzaniaDźwięku|UpłynięcieCzasu”
        czasUtrzymywaniaReakcji="1000" odłóżPrzejścieDoScenyDoZakończeniaReakcji="tak|nie"
        automatycznaReakcjaPoCzasie="1000" typAkcji="Brak|Ramka|Przesunięcie" grubośćRamki="10"
        kolorRamki="Red" nazwaRysunku="zoe_purple" nazwaDźwięku="zoe_dzwiek" ścieżka="0,-425;-500,0"
        szybkość="10" wyłączPoZakończeniu="tak|tekst="GIML" czcionka="Times" rozmiarCzcionki="30"
        stylCzcionki="ibus"znacznik="nazwa znacznika" znacznikOpóźniony="nazwaznacznikaopóźnionego"
        opóźnienieZnacznikaOpóźnionego="2000"nazwaObszaruWłączanegoPoRozpoczęciu="prim;bis"
        nazwaObiektuWłączanegoPoZakończeniu="prim;bis"
        nazwaObszaruWyłączanegoPoZakończeniu="koło2;bis"
        typAnimacji="Brak|Powiększanie|ObrotyWLewo|ObrotyWPrawo|KołysanieWPoziomie|KołysanieWPionie"
        okresAnimacji="200" amplitudaAnimacji="0,1" klawiszReakcji="r"
        poAktywacji="minusJeden_poAktywacji" poRozpoczęciuReakcji="minusJeden_poRozpoczęciuReakcji"
        poZakończeniuReakcji="minusJeden_poZakończeniuReakcji"
        poPowrocieDoStanuNormalnego="minusJeden_poPowrocieDoStanuNormalnego"
        poZmianieStanu="minusJeden_poZmianieStanu">
    <aktywacja te same atrybuty, co w reakcji/>
    <reakcja typAkcji="Brak|Ramka|PrzejścieDoSceny|Przesunięcie" grubośćRamki="10" kolorRamki="Blue"
             nazwaRysunku="cookie_monster" nazwaDźwięku="chime"ścieżka="0,-425;-500,0" szybkość="5"
             wyłączPoZakończeniu="tak" nazwaObszaruWłączanegoPoRozpoczęciu="prim;"
             nazwaObszaruWłączanegoPoZakończeniu="prim;bis"
             typAnimacji="Brak|Powiększanie|ObrotyWLewo|ObrotyWPrawo|KołysanieWPoziomie|KołysanieWPionie"
             okresAnimacji="200" amplitudaAnimacji="0,1"nazwaDocelowejSceny="2" />
</obszar>
```

3.7 Przykłady użycia i perspektywy rozwoju

Przedstawione powyżej elementy zdefiniowane w GIML mają być proste w implementacji, a oferowane możliwości są ograniczone tylko wyobraźnią twórcy. Autor języka otworzył nowe perspektywy dla osób, które nie są programistami, dając im możliwość tworzenia aplikacji sterowanych wzrokiem. Rezultaty tego podejścia są już zauważalne podczas prowadzonych w ICNT badań psychologicznych. Opisujące je pliki GIML zostały przygotowane przez osoby nie mające dużego doświadczenia w programowaniu. Prosta składnia i łatwość użycia znaczników w plikach XML dały możliwość samodzielnego przygotowania eksperymentów przez osoby, które je przeprowadzają. Przykład prostego użycia języka GIML, wraz z wyróżnieniem w nim poszczególnych znaczników przedstawia rys. 3.3.



Rys. 3.3 Przykład pliku XML z użyciem języka GIML wraz z wyróżnieniem składni języka.

Pierwszym interpreterem języka GIML jest platforma GCAF. Nowy sposób budowy aplikacji wydaje się mieć duże możliwości rozwoju, szczególnie w tych sektorach rynku, gdzie wykorzystuje się zindywidualizowane oprogramowania sterowane wzrokiem i ukierunkowane na konkretne funkcjonalności. GIML może być użyty również do częściowej zmian wyglądu aplikacji (np. motywu) przez samych użytkowników, co pozwoliłoby na większą indywidualizację i lepsze dopasowanie aplikacji do własnych potrzeb.

4. Platforma GCAF

4.1 Wymagania aplikacji

- System operacyjny: Windows 7 SP1 (x64, x86) lub wyższy
- Pamięć operacyjna: 1024 MB wolnej pamięci
- Pamięć twarda: około 500 MB wolnej przestrzeni dyskowej
- Interfejsy: port RJ-45 wraz z zainstalowanym sterownikiem do karty sieciowej, USB w wersji 2.0 lub wyższej
- Oprogramowanie: .NET 4.5, biblioteka standardowych kodeków audio i filmów
- Rozdzielczość ekranu: minimum 1280 × 1024, format 16:9, 21:9

4.2 Opis aplikacji

GCAF, czyli Gaze Controlled Applications Framework jest platformą, która służy do tworzenia własnych aplikacji, które są sterowane za pomocą wzroku. Taka forma komunikacji jest możliwa, dzięki wykorzystaniu eyetrackerów. To nowatorskie podejście sprawia, że program znajduje zastosowania w nowych dziedzinach nauki oraz rynku. Ponadto użycie wzroku jako urządzenia wskazującego pozwala dotrzeć do szerszego grona odbiorców, m.in. dzieci oraz osób niepełnosprawnych, dla których wzrok jest jedyną formą interakcji z otoczeniem. Możliwość tworzenia własnych aplikacji sterowanych wzrokiem uzyskano dzięki wykorzystaniu języka GIML do opisu UI, czyli wyglądu aplikacji, oraz jej reakcji na spojrzenie. Odpowiednio przygotowane pliki XML są w czasie rzeczywistym przetwarzane na obraz wyświetlany użytkownikowi. GCAF implementuje wszystkie możliwości oferowane przez GIML. Użytkownik ma pełną swobodę w projektowaniu i używaniu stworzonej przez siebie aplikacji.

Twórcami platformy GCAF są: Jacek Matulewski, który jest kierownikiem projektu oraz głównym programistą GCAF i twórcą języka GIML; Bibianna Bałaj prowadząca nadzór merytoryczny, testy oraz badania z wykorzystaniem eyetrackera; Rafał Linowiecki

- drugi programista, zajmujący się również badaniami i rozwojem produktu; Alicja Majka odpowiadająca za pokrycie aplikacji testami jednostkowymi oraz Agnieszka Ignaczewska, która dokonuje testów aplikacji GCAF i przeprowadza badania również z jej użyciem. Mój udział w tworzeniu platformy GCAF rozpoczął się od wdrożenia obsługi dźwięku oraz filmów, przy użyciu zewnętrznych bibliotek. Zajmowałem się również obsługą nagrywania filmu z przebiegu badania. Jestem odpowiedzialny za statystyki obszarów tworzone podczas programu, generowanie plików XML ze strukturą obszarów zainteresowania, która jest interpretowana przez serwer ET. Podczas pracy nad GCAF zaimplementowałem sterownik do eyetrackera firmy Mirametrix, który korzysta z standardu stworzonego na potrzeby eyetrackingu, mianowicie interfejsu Open Eye-gaze Interface. Dokonywałem pierwszych pokryć testami jednostkowymi i integracyjnymi programu. Zajmowałem się również promocją produktu podczas konferencji eyetrackingowej, wystąpień m.in. w siedzibie Microsoft, na Politechnice Gdańskiej oraz spotkań z inwestorami.

Platforma oferuje możliwość rejestracji przebiegu badania. Analiza, która jest przeprowadzana podczas trwania badania, a następnie wyświetlana po zakończeniu programu, pozwala zastąpić bardzo kosztowne programy specjalistyczne przeprowadzające różnego rodzaju analizy ruchów oka. Dzięki temu bardzo łatwo jest przygotować program do konkretnego badania, który od razu po jego zakończeniu dostarcza interesujących wyników.

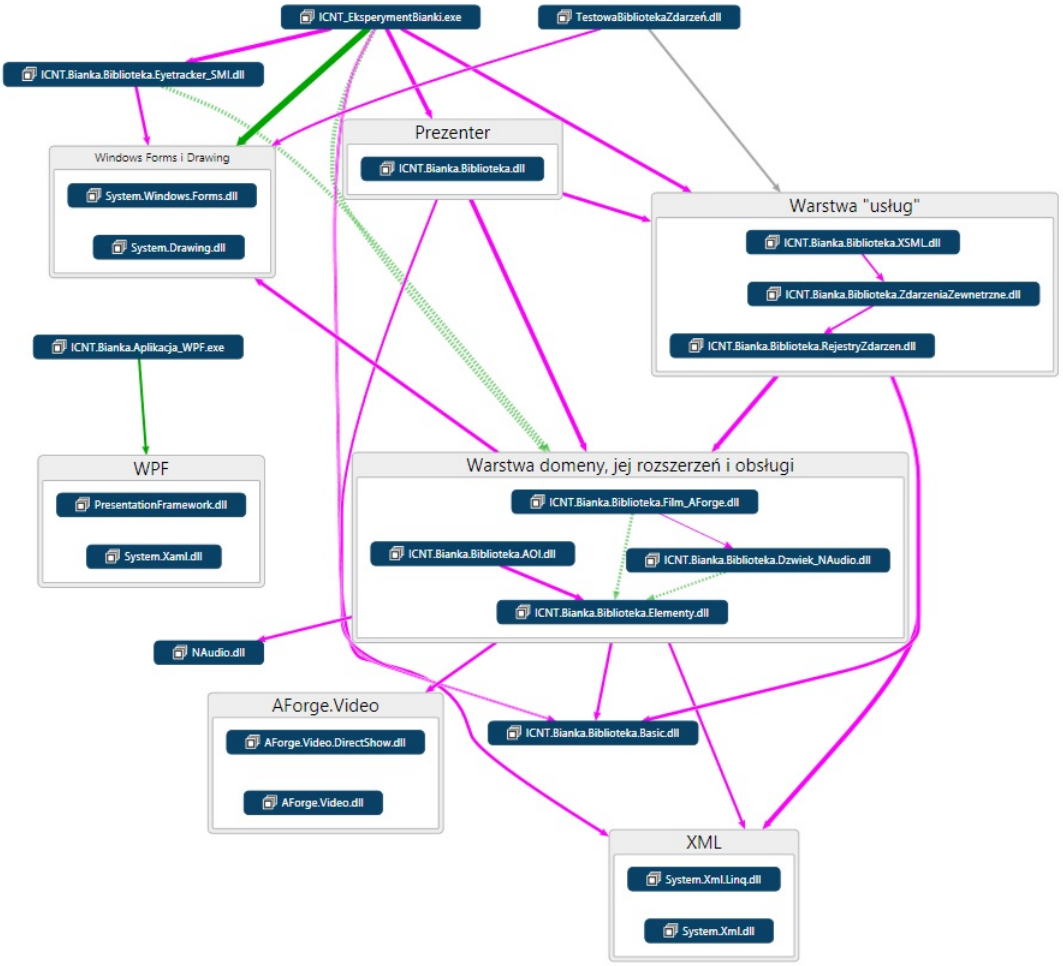
Rys. 4.1 Logo platformy GCAF

Aplikacja została zaimplementowana przy użyciu platformy .NET w języku C# przy wykorzystaniu narzędzi deweloperskich firmy Microsoft. Architektura aplikacji opiera się na wzorcu wielowarstwowym (rys. 4.2), co pozwala zachować standardy tworzenia rozbudowanych aplikacji, jak również przejrzystość logiki kodu źródłowego. Cała aplikacja

została pokryta testami jednostkowymi oraz integracyjnymi, co pozytywnie wpływa na stabilność działania całej platformy i weryfikację pojawiających się błędów już na etapie tworzenia kodu źródłowego. To sprawia, że klient otrzymuje aplikacje, która działa stabilnie. Ważną cechą GCAF jest jej modułowość - każda z funkcjonalności jest zaimplementowana w osobnych bibliotekach. Taki zabieg ma służyć odpowiedniemu dopasowaniu możliwości aplikacji dla konkretnego klienta - jeśli klientem ma być osoba indywidualna, której zależy tylko na aplikacji sterowanej wzrokiem, bez konieczności zbierania informacji i rejestrowania przebiegu działania programu, to nie ma konieczności sprzedawania oprogramowania z całą funkcjonalnością. Zabieg ten ma na celu zróżnicowanie oferowanych możliwości programu w zależności od grupy odbiorców, co pozwoli na korzystniejsze dopasowanie ceny do funkcjonalności oprogramowania - klient płaci tylko za to, czego potrzebuje. Przewagą w porównaniu do innych aplikacji jest możliwość użycia języka GIML do stworzenia własnego programu, która jest dostępna w każdym z oferowanych pakietów GCAF.

Obecnie GCAF wspiera dwa rodzaje eyetrackerów - firmy SMI oraz Mirametrix. W przypadku Mirametrixa zaimprementowana jest obsługa otwartego standardu, który można wykorzystać do sterowania eyetrackerami firmy Tobi oraz innych, które nie zostały jeszcze przebadane. Eyetracker SMI, ze względu na jego precyzyjność pomiarów, jest przeznaczony raczej do specjalistycznych zastosowań, a co za tym idzie wysoki koszt (250 tys.). Są one obecnie wykorzystywane w ośrodkach badawczych podczas prowadzonych eksperymentów, m.in. w ICNT. Drugi rodzaj eyetrackerów - firmy Mirametrix to przykład niewielkich rozmiarów urządzenia, który oferuje dość dobrą precyzję przy stosunkowo niewielkiej cenie (4-8 tys.). Jego główną zaletą jest łatwość użycia i duża mobilność. Planuje się wsparcie innych eyetrackerów przez GCAF w jej kolejnych wersjach.

GCAF jest kierowany do szerokiego grona odbiorców z różnym poziomem doświadczenia w korzystaniu z komputerów. Działanie aplikacji można podzielić na część konfiguracyjną, aplikację z użyciem GIML oraz zwracane wyniki. Pierwszą z nich jest okno konfiguracyjne, które dostarcza użytkownikowi możliwości wprowadzenia własnych ustawień dotyczących działania głównej części aplikacji. W części głównej aplikacji pojawia się interfejs użytkownika, stworzony przez użytkownika z użyciem języka GIML, którego sterowanie odbywa się przy pomocy eyetrackera lub dowolnego urządzenia wskazującego, rozpoznanego przez system operacyjny, który może imitować eyetracker - może to być



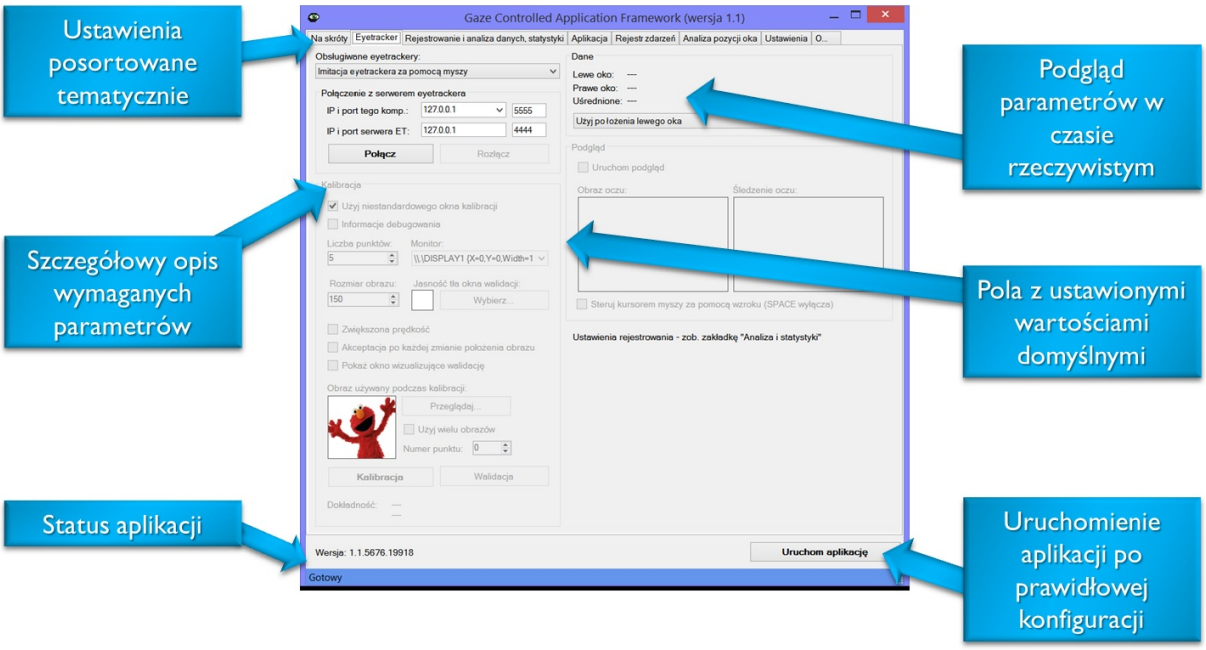
Rys. 4.2 Wielowarstwowa architektura platformy GCAF

mysz. Po zakończeniu działania głównej części programu, zwracane są dane dotyczące jej przebiegu. Oczywiście ta część dotyczy tylko tych wersji GCAF, które posiadają funkcjonalność zbierania i analizowania zachowania wzroku użytkownika. Dokładniejszy opis poszczególnych modułów programu zostały przedstawione w kolejnych podrozdziałach pracy.

4.3 Okno konfiguracyjne

Ważną funkcjonalnością aplikacji sterowanych wzrokiem jest ich możliwość dostosowania do potrzeb użytkownika. W przypadku platformy GCAF pozwala na to okno konfiguracyjne. Jest to tym ważniejsze, że użytkownik sam może tworzyć aplikacje.

Wygląd okna konfiguracyjnego został zaprojektowany z myślą intuicyjnego rozmieszczenia obiektów UI oraz uporządkowania dostępnych ustawień według zagadnień, których



Rys. 4.3 Okno konfiguracyjne GCAF z opisem głównych jego elementów

dotyczą. W tym celu okno zawiera zakładki, na których znajdują się parametry dotyczące eyetrackerów, rejestru zdarzeń aplikacji i wyglądu okna aplikacji. Okno konfiguracyjne z ustawieniami eyetrackera przedstawia rys. 4.3. Okno konfiguracyjne GCAF można utożsamiać z panelem sterującym działanie całej aplikacji.

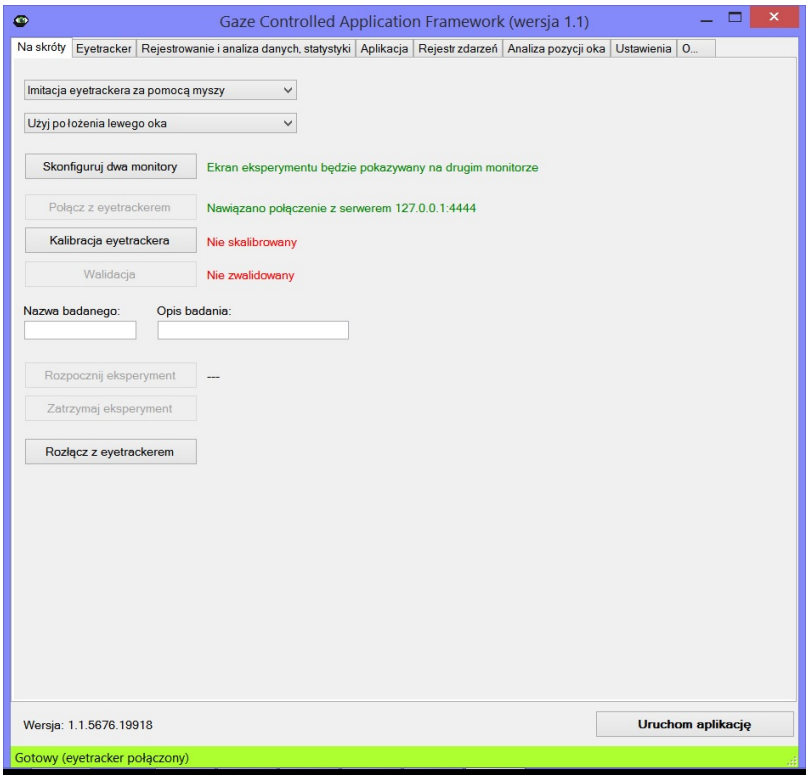
4.3.1 Zakładka *Na skróty*

Dla łatwiejszego zarządzania programem wszystkie najważniejsze elementy konfiguracji zostały umieszczone w zakładce *Na skróty*, która pokazuje się jako pierwsza po uruchomieniu programu. Mniejsza liczba elementów i przejrzystość ich rozmieszczenia powodują, że osoba, która wykonuje badanie, bardzo szybko może uruchomić program służący do eksperymentu bez konieczności przeszukiwania interesujących parametrów po wszystkich zakładkach ekranu z ustawieniami dotyczącymi eyetrackera. Zakładka ta umożliwia m.in. wybór urządzenia wskazującego spośród wspieranych eyetrackerów lub też urządzenia, które pozwala imitować eyetracker. To drugie rozwiązanie pozwala korzystać z aplikacji również tym klientom, którzy nie potrzebują w swoich programach wykorzystania wzroku do ich sterowania. Można w niej także przygotować kod GIML bez dostępu do eyetrackera. Kolejnym ustawieniem jest wybór oka, którego ruch ma być analizowany przez eyetracker. Są trzy możliwości: oko lewe, prawe lub ich uśrednione położenie. Różnica polega na tym,

że każdy z nas posiada tzw. oko dominujące, które odpowiada za to co i w jakim miejscu widzimy. Drugie oko natomiast jest pomocnicze a patrzenie tylko przez nie powoduje, że punkt wzroku się przesuwa w odniesieniu do patrzenia wraz z użyciem oka dominującego. Często jednak uśrednia się oba położenia oczu, które dobrze przybliża położenie wzroku przy użyciu samego oka dominującego. Dalsze ustawienia to możliwość skonfigurowania dwóch monitorów. Wówczas właściwe okno aplikacji z interfejsem stworzonym przez użytkownika jest uruchamiane na drugim monitorze z tej zakładki. Można również połączyć się z eyetrackerem - każdy z obsługiwanych eyetrackerów łączy się przez kabel sieciowy z wykorzystaniem protokołu TCP/IP. W tym celu konieczne staje się podanie zarówno adresu IP i portu eyetrackera oraz komputera, na którym uruchomiono GCAF. Szczegółowe parametry połączenia ustala się w zakładce *Eyetracker*, która przedstawiona zostanie w kolejnym kroku. Pozostałe parametry, takie jak kalibracja oraz walidacja są wymagane do poprawnego działania okulografu. Proces walidacji pozwala uzyskać informację o poprawności przebiegu procesu kalibracji. Ostatnie parametry dostępne w tej zakładce dotyczą bezpośrednio eksperymentu. Podaje się tutaj nazwę osoby badanej oraz opis badania, które są umieszczane w zapisywanych przez program wynikach. Z tego poziomu można również rozpocząć oraz zatrzymać eksperyment a także połączyć lub rozłączyć eyetracker. Wygląd zakładki *Na skróty* przedstawia rys. 4.4.

4.3.2 Zakładka *Eyetracker*

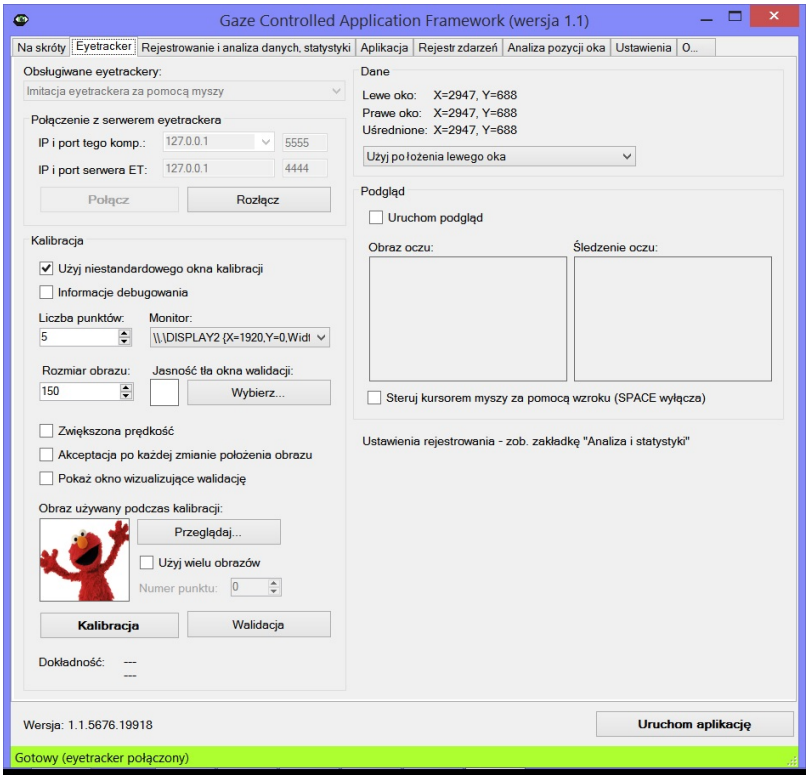
Zakładka *Eyetracker* (rys. 4.5) umożliwia dostosowanie parametrów związanych z działaniem okulografu. Dla przejrzystości parametry pogrupowano w bloki odpowiadające różnym funkcjom obsługi eyetrackera. Ustawienie, które dostępne jest również w zakładce *Na skróty* pozwala na wybór urządzenia wskazującego. Po dokonaniu wyboru należy skonfigurować parametry połączenia z obszaru *Połączenie z serwerem eyetrackera*, w których należy podać adresy IP serwera i klienta (zob. rys. 2.8). Po uzupełnieniu tych pól można kliknąć przycisk *Połącz* aby połączyć klienta z eyetrackerem. Należy pamiętać o odpowiednim ustawieniu firewall'a, który pozwoli na wymianę informacji pomiędzy serwerem w sieci. Po uzyskaniu połączenia pojawi się powiadomienie lub informacja o problemie, jeśli taki wystąpi. Kolejna część okna odpowiada za proces kalibracji urządzenia. Istnieje możliwość wyboru własnego ekranu kalibracji bądź też skorzystania z kalibracji przy użyciu oprogramowania dostarczanego wraz z eyetrackerem.



Rys. 4.4 Okno konfiguracyjne GCAF z wybraną zakładką *Na skróty* pozwalającą dokonać szybkich ustawień eksperymentu

Informacje debugowania, które można wybrać służą wyświetleniu pomocniczych informacji odnośnie stanu działania aplikacji podczas procesu kalibracji. Parametr liczba punktów określa, na podstawie ilu punktów wyświetlonych na ekranie będzie kalibrowany okulograf. Ważne tutaj jest, że tylko niektóre liczby punktów są poprawne, np. w przypadku SMI są to 5, 7, 9 oraz 11. Istnieje możliwość wyboru monitora, na którym zostanie wyświetlone okno kalibracji. Badania w ICNT pokazały, że w przypadku kalibracji małych dzieci występuje problem z ich skupieniem na punkcie. Dlatego rozszerzono możliwości okna kalibracji o wyświetlanie obrazków, np. postaci z bajek, które mają przyciągnąć uwagę dziecka, zamiast punktów. GCAF umożliwia wybór nie tylko jednego, ale również kilku obrazków w jednym procesie kalibracji, przypisując konkretny obrazek do danego punktu kalibracji. Możliwe jest określenie ich rozmiarów oraz tła na którym będą się znajdować. Pozostałe parametry to: zwiększona prędkość, która przyspiesza proces kalibracji (polecana w przypadku doświadczonych użytkowników), akceptacja po każdej zmianie położenia obszaru, która ułatwia kalibrację np. dzieci, oraz prezentacja okno wizualizującego walidację. W panelu *Kalibracja* znajdują się także pola, które po walidacji prezentuje informację o dokładności kalibracji. Informacje o bieżącym położeniu punktu

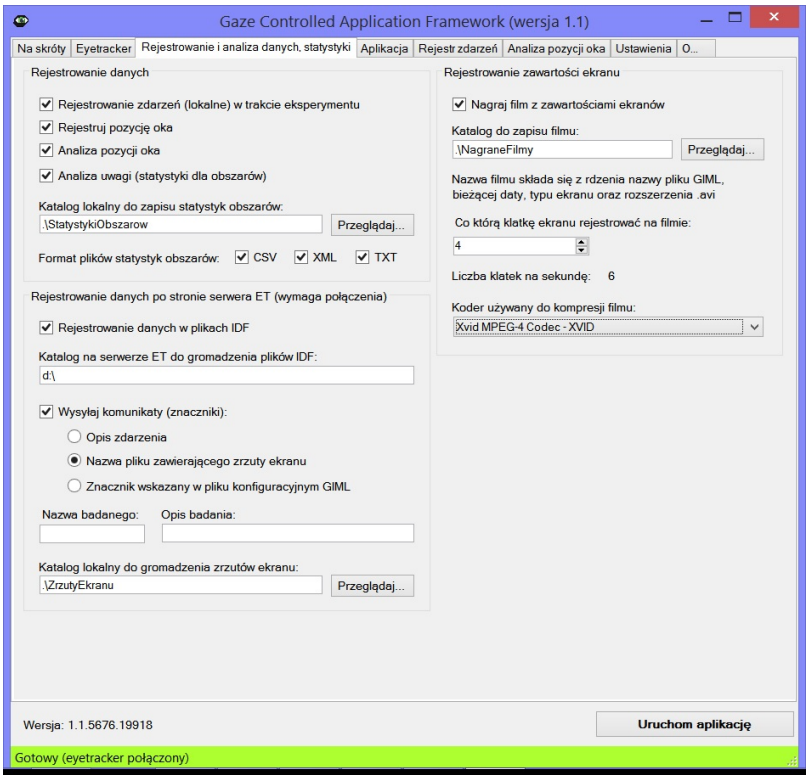
wzroku są wyświetlane w kolejnym obszarze zakładki *Eyetracker* - obszarze *Dane*. Ostatnim panelem jest *Podgląd*, który na bieżąco pokazuje obraz z eyetrackera zawierający podgląd oczu oraz położeniue oczu na obrazie kamery eyetrackera. Ciekawostką ustawień jest możliwość włączenia sterowania kursorem myszy za pomocą wzroku, czyli możliwość użycia okulografu jako urządzenia sterującego.



Rys. 4.5 Okno konfiguracyjne GCAF z wybraną zakładką *Eyetracker* pozwalającą wprowadzić szczegółowe ustawienia działania okulografu

4.3.3 Zakładka *Rejestrowanie i analiza danych, statystyki*

Następna zakładka *Rejestrowanie i analiza danych, statystyki* (rys. 4.6) jest przeznaczona dla osób, wykorzystujących GCAF do badań. Podobnie jak w poprzedniej zakładce, również tutaj ustawienia zostały pogrupowane tematycznie. Pierwszym panelem jest *Rejestracja danych*, którego pola pozwalają na wybór sposobu zapisu danych po przeprowadzeniu badania. Poza rejestracją występujących zdarzeń oraz pozycji oka, GCAF oferuje możliwość przeprowadzania ich analizy oraz analizy uwagi. Ta ostatnia traktuje obszary zdefiniowane w GIML, jako obszary zainteresowania (AOI, ang. *Array of Interests*). Tworzony jest wówczas plik XML z opisem obszarów AOI, który jest wykorzystywany w ana-



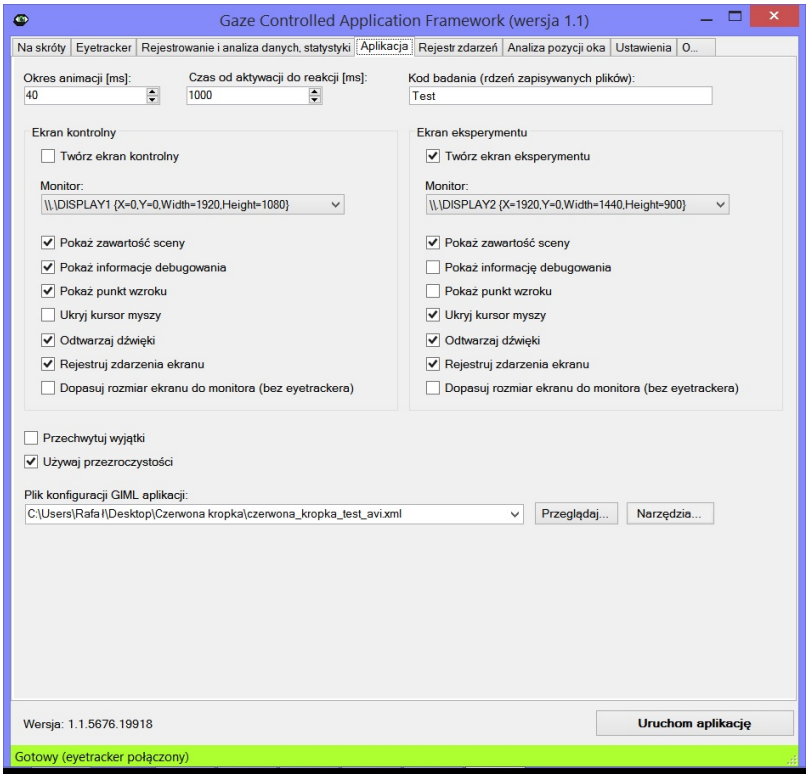
Rys. 4.6 Okno konfiguracyjne GCAF z wybraną zakładką *Rejestrowanie i analiza danych, statystyki* dotyczącą ustawień rejestracji i analizy działania aplikacji

lizie przez oprogramowanie BeGaze. AOI są standardową koncepcją przy opisie obiektów, które podlegają analizie. Uzyskane w ten sposób statystyki mogą być są zapisywane do plików z rozszerzeniem CSV, XML lub TXT co umożliwia ich łatwy import do programów zajmujących się ich dalszą obróbką. Można wybrać katalog, w którym pliki mają być zapisywane. Drugi panel omawianej zakładki to ustawienia odnośnie rejestracji danych po stronie serwera eyetrackera. Zbierane dane pochodzące z okulografu są zapisywane na serwerze ET w specjalnym formacie IDF, który później jest analizowany przy użyciu specjalistycznych programów. Kolejne parametry dotyczą wysyłanych znaczników, które były już wcześniej omawiane przy okazji opisu języka GIML i służą one do oznaczenia obszarów, które później identyfikują w plikach IDF. Jest to pomocne przy późniejszej analizie w innych programach. Ustawienia nazwy badanego oraz opisu badania są tymi samymi, które były omawiane już przy zakładce *Na skróty*. Nową możliwością jest wybór katalogu do gromadzenia zrzutów ekranu. GCAF oferuje użytkownikowi możliwość rejestracji działania okna aplikacji, z własnym interfejsem użytkownika, w zapisywanych po każdej zmianie stanu sekwencji zrzutów ekranu, które ułatwiają późniejszą analizę np. w oprogramowaniu BeGaze. Możliwe jest również nagranie filmu z działania aplikacji do formatu

AVI (oferując przy tym zapis wybiórczy, co którychś zrzut ekranu). Ponieważ prowadzone badania są często czasochłonne a nagrywanie filmów zabiera dużo zasobów pamięciowych, GCAF oferuje wybór kodera, którym kompresowany jest nagrywany film. Wykorzystano tutaj także zapis w czasie rzeczywistym, dzięki czemu po zakończeniu aplikacji film jest od razu dostępny dla użytkownika, a obraz z nagrania nie jest zbędnie przechowywany w pamięci RAM.

4.3.4 Zakładka *Aplikacja*

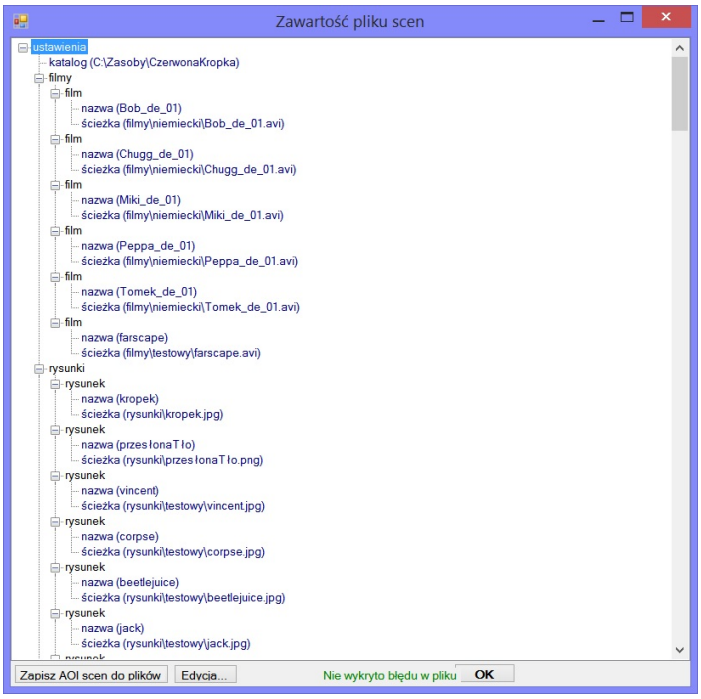
Najważniejszą grupą ustawień, dotyczących działania aplikacji GCAF dostępna jest w zakładce *Aplikacja* (rys. 4.7). Parametr *Okres animacji* określa czas (w ms), który potrzebny jest na odtworzenie animacji. Wspomniane podczas opisu języka GIML stany aktywacji i reakcji są ustawiane przez określenie czasu, po którym stan aktywacja przechodzi w stan reakcji. Kod badania, będący kolejnym parametrem ustawień jest wartością, która identyfikuje przeprowadzone badanie. Zakładka *Aplikacja* zawiera również dwa pola



Rys. 4.7 Okno konfiguracyjne GCAF z wybraną zakładką *Aplikacja* oferującą bezpośrednią konfigurację ustawień uruchamianej aplikacji

posiadająca te same możliwości ustawień, ale dotyczą monitora kontrolnego i monitora

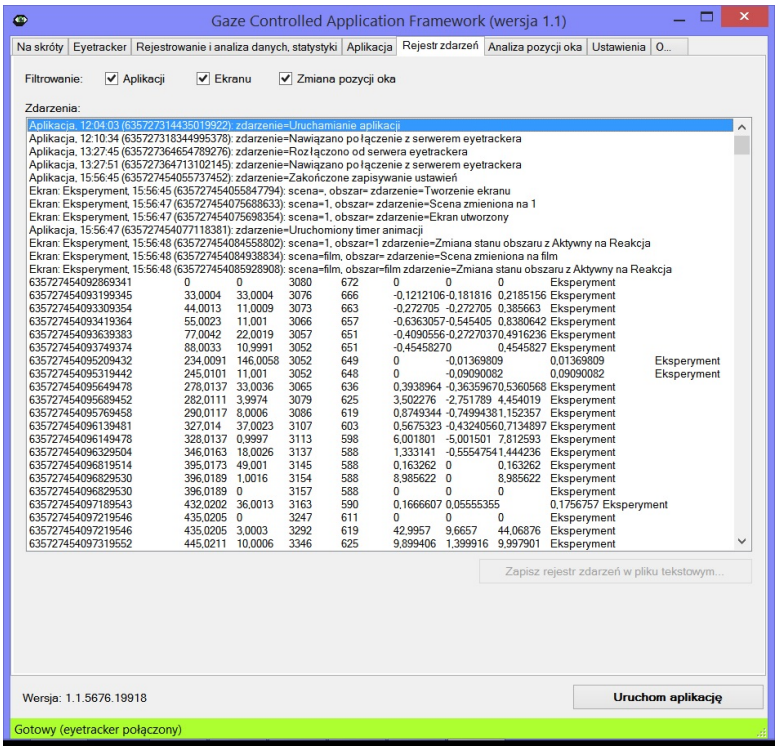
eksperymentu. Pierwszy z nich jest przeznaczony do osoby przeprowadzającej badanie. Dzięki niemu ma ona kontrolę nad tym, co aktualnie jest wyświetlane na ekranie eksperymentu, czyli co widzi osoba badana. Funkcjonalność ta jest oczywiście możliwa tylko, gdy korzysta się z co najmniej dwóch monitorów. Na każdym z ekranów jest możliwość wyświetlenia zawartości sceny, czyli obiektów, które zostały opisane w pliku GIML. Podczas wyświetlania tych obiektów można włączyć wyświetlanie informacji debugowania czyli parametrów opisujących stan danego obszaru ale również stan sceny. Informacje te zawierają dane o tym, w jakim stanie znajduje się dany obiekt (nieaktywny, aktywacja, reakcja). Ustawienie *Pokaż punkt wzroku* pozwala wyświetlać spojrzenia. GCAF umożliwia również wyłączenie odtwarzania dźwięku czy ukrycie kursora myszy podczas badania. Parametr *Dopasuj rozmiar ekranu do monitora* pozwala na zastosowanie tzw. full screna w przypadku, kiedy rozmiar ekranu w pliku GIML różni się od rozdzielczości monitora, na którym ma zostać wyświetlony interfejs użytkownika. Dla zastosowań developerskich dodano możliwość przechwytywania wyjątków i wyświetlania komunikatów. Opcja *Używaj przezroczystości* pozwala na uruchomienie kanału alfa dla wyświetlanych obiektów. Wyświetlanie okna aplikacji z własnym interfejsem jest możliwe po załadowaniu pliku GIML w polu *Plik konfiguracji GIML aplikacji* właśnie w zakładce *Aplikacja*. Wybiera się wówczas ścieżkę pliku XML z kodem w języku GIML. Istnieje możliwość weryfikacji załadowanego pliku GIML - służy do tego przycisk *Narzędzia...*. Po jego wybraniu ukazuje się okno wraz z załadowaną strukturą pliku GIML (rys. 4.8). Służy ono do wskazania ewentualnych błędów dotyczących zarówno składni języka, jak i poprawnych odwołań do zdefiniowanych zasobów. Z tego poziomu można również przejść do edycji pliku w zewnętrznym programie oraz wyeksportować zdefiniowane obszary jako obszary zainteresowania AOI. Narzędzie to umożliwia szybką weryfikację, które elementy są poprawne (kolor niebieski), a które nie (kolor czerwony). W przypadku niepoprawnych elementów, po najechaniu na nie kursorem myszy pojawia się informacja o błędzie. Jeśli u dołu ekranu pojawi się zielony napis *Nie wykryto błędu...* oznacza to, że cały plik poprawnie przeszedł proces walidacji i można go uruchomić. Jeśli pojawi się czerwony napis *Wykryto błąd w pliku* to oznacza, że wystąpił błąd i aplikacja nie zostanie uruchomiona z takim plikiem.



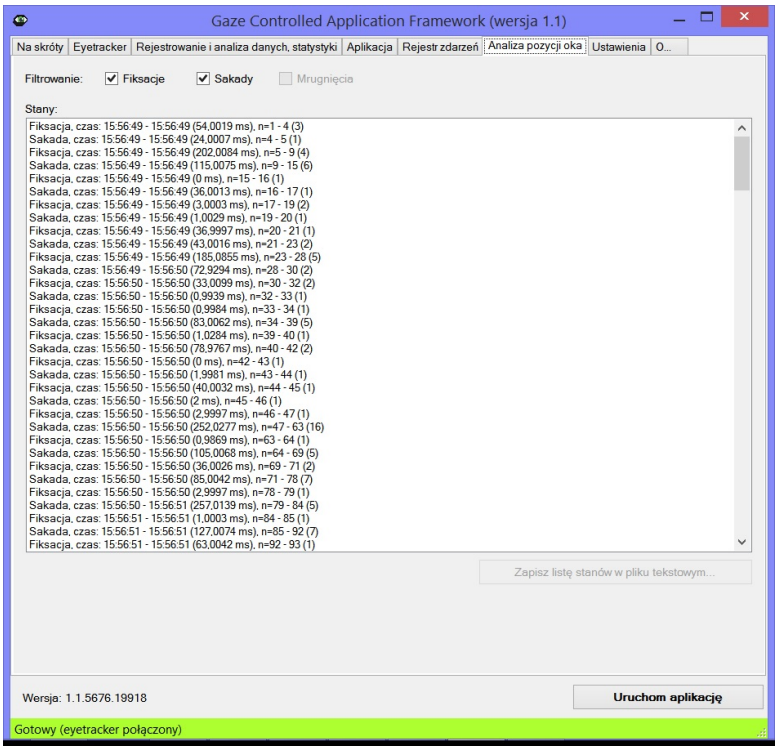
Rys. 4.8 Okno walidacji plików GIML oferującym podgląd całej struktury interfejsu aplikacji

4.3.5 Zakładki *Rejestr zdarzeń* i *Analiza pozycji oka*

Okno konfiguracyjne nie tylko oferuje możliwość kontroli ustawień, ale odpowiada także za podgląd tego, co dzieje się z aplikacją. Informacje o tyn umieszczone są w dwóch kolejnych zakładkach: *Rejestr zdarzeń* i *Analiza pozycji oka*. Przedstawiają one dane, które w czasie rzeczywistym są aktualizowane przez aplikację. Pierwsza z zakładek odpowiada za podgląd zdarzeń aplikacji, ekranu oraz pozycji oka, czyli przebiegu badania. Przykład zwracanych danych przedstawia rys. 4.9. Druga prezentuje wynik analizy danych oka uzyskiwanych z eyetrackera. Przedstawia parametry fiksacji oraz sakkad (rys. 4.5). W przyszłej wersji aplikacji planowane jest również wykrywanie mrugnięć, o których informacja będzie się pojawiać na tej zakładce.



Rys. 4.9 Okno konfiguracyjne GCAF z wybraną zakładką *Rejestr zdarzeń* zawierającą przykładowe dane z uruchomionej aplikacji



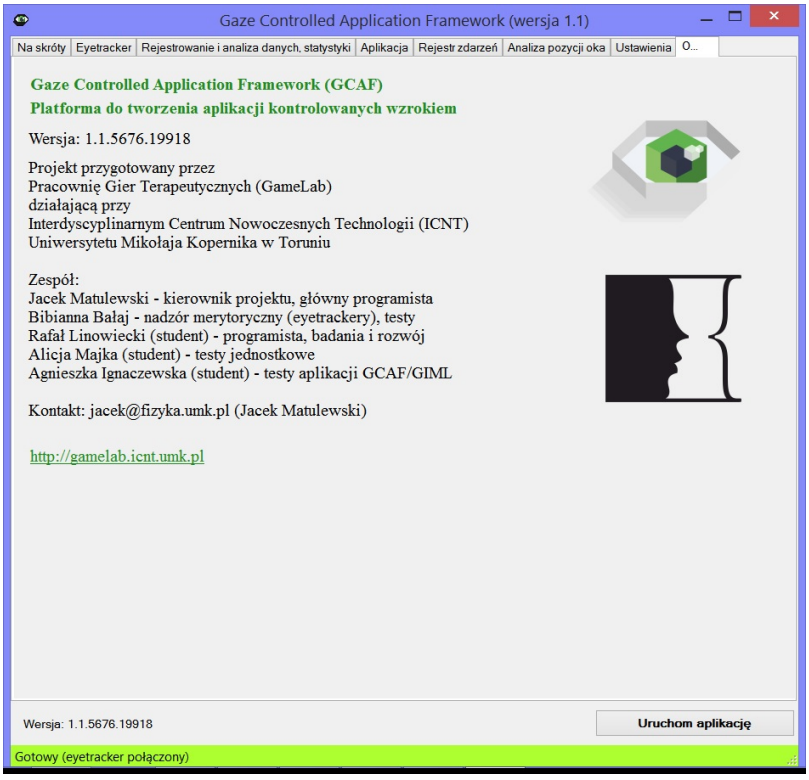
Rys. 4.10 Okno konfiguracyjne GCAF z wybraną zakładką *Analiza pozycji oka* wraz z przykładowymi danymi o fiskacjach oraz sakkadach podczas badania

4.3.6 Zakładka *Ustawienia*

Zakładka *Ustawienia* oferuje zmianę języka interfejsu. Obecnie wspierane są dwa języki: polski oraz angielski. W przyszłej wersji planowana jest rozbudowa o język niemiecki oraz francuski. Drugi z parametrów daje możliwość wyświetlania kolorowego tła na pozostałych monitorach (pomijając ten, na którym wyświetlana jest scena). Jest to domyślnie stosowane podczas badania, aby nie rozpraszać osoby badanej.

4.3.7 Zakładka *O...*

Ostatnia z zakładek etapu konfiguracji jest poświęcona informacji o platformie GCAF. Zawiera informacje o wersji, pomysłodawcy projektu, jej twórcach oraz danych kontaktowych (rys. 4.11).



Rys. 4.11 Okno konfiguracyjne GCAF z wybraną zakładką *O...* zawierająca informację o platformie GCAF

4.4 Działanie programu i sterowanie wzrokiem

Kolejnym etapem działania platformy GCAF jest uruchomione okno z interfejsem utworzonym w oparciu o plik GIML. Cały pomysł opiera się na tym, że brak tutaj sztywno określonych reguł jak ma on wyglądać i jakie funkcje spełniać. Zależy to jedynie od umiejętności użytkownika posługiwania się językiem GIML i oczywiście tego, co ma aplikacja "robić". Okno to jest uruchamiane z ekranu konfiguracyjnego, po wcześniejszym ustawieniu parametrów poprawnych wczytaniu pliku GIML.

Działanie aplikacji rozpoczyna się po przyciśnięciu przycisku *Uruchom aplikację*, znajdującym się na dolnym pasku okna konfiguracyjnego. Podczas uruchamiania odczytywany jest plik GIML, który jest parsowany. Sprawdzane są wszystkie użyte parametry i znaczniki pod kątem ich zgodności ze standardem. Jeśli zostało wprowadzone w pliku odwołanie do zasobu, którego fizycznie nie ma na dysku, program zgłosi błąd. Pomimo dużej złożoności parsowanie odbywa się bardzo szybko.

Kolejny etap odpowiada za wyświetlenie domyślnej sceny. Z wczytanych obiektów pobierane są te, które dotyczą sceny domyślnej a następnie zostają wyświetlone na ekranie. Wszystkie ich parametry, czyli m.in. rozmiar, kształt, położenie są określone w pliku GIML. Dotyczy to także zdarzeń zewnętrznych i animacji. Wszystkie zasoby wymagane w scenie domyślnej są ładowane z zewnętrznych plików i wyświetlane (rysunki) lub uruchamiane (dźwięki i filmy) zgodnie z atrybutami w GIML. Do odtwarzania dźwięków platforma GCAF wykorzystuje zewnętrzną bibliotekę NAudio, dystybuowanej na licencji Open Source. Jej użycie pozwala na odtwarzanie popularnych formatów dźwięku przy użyciu kodeków systemowych. Odtwarzanie odbywa się w oddzielnych wątkach, przez co działanie całej aplikacji nie jest spowalniane. Ponadto oferuje ona bogaty pakiet operacji na dźwiękach. Do filmów natomiast użyto biblioteki AForge, również na licencji Open Source. Odtwarzane są filmy w formacie AVI. Sposób, w jaki biblioteka ta przetwarza strumień wideo na obraz jest kompatybilny z obiektami, które GCAF wyświetla podczas tworzenia scen. Dzięki temu użycie tej biblioteki nie zmienia logiki całego programu, w szczególności przygotowanych zrzutów ekranu. Ważny jest fakt, iż jakoś odtwarzania przy użyciu różnych kodeków jest bardzo dobra, a wykorzystywanie pamięć jest w odpowiednich momentach zwalniana. Odtwarzanie filmów jest również realizowane w osobnych wątkach.

Po opisanych wcześniej procesach wyświetlana jest użytkownikowi pierwsza (domyślna) scena wraz z obsługą przy użyciu wybranego urządzenia. Jeśli podczas konfiguracji zostało wybrane rejestrowanie danych aplikacji, to w osobnych wątkach są one przetwarzane i zapisywane oraz wyświetlane na bieżąco w zakładkach *Rejestracja zdarzeń* oraz *Analiza pozycji oka* okna konfiguracyjnego. W przypadku zapisu sekwencji zrzutów ekranu, osobny proces co pewien ustalony czas uruchamia metodę, która przechwytuje obraz i zapisuje go od razu do pliku - nie ma potrzeby trzymania go w pamięci. Istnieje także możliwość nagrania przebiegu działania aplikacji do filmu w formacie AVI. W tym przypadku również jest wykorzystywana biblioteka AForge, ale korzystająca z obiektów odpowiedzialnych za zapis strumienia wideo. Istnieje także możliwość bieżącej kompresji strumienia wideo, co zdecydowanie ogranicza rozmiar uzyskanego w ten sposób pliku. Przeprowadziłem testy wydajnościowe w przypadku zapisu bez kompresji i z kompresją. W pierwszym przypadku zużycie zasobów procesora było niższe niż z użyciem kompresji, jednak zużycie pamięci znacznie wyższe. Ponieważ proces nagrywania działa w osobnych wątkach (w tle), różnica w użyciu procesora jest niezauważalna dla użytkownika aplikacji. Do niego więc należy wybór, czy chce mieć dużych rozmiarów film kosztem niższego zużycia procesora, czy poświęcić trochę więcej zasobów w zamian za mały rozmiar filmu. Lepsze wydaje się drugie rozwiązanie, które również jest wykorzystywane podczas aktualnie prowadzonych badań w ICNT.

Oprócz zbierania i rejestrowania danych, program na bieżąco dokonuje także ich analizy, których wyniki zapisuje do plików. Aplikacja została pod tym względem zoptymalizowana, aby w jak najmniejszym stopniu wykorzystywać zasoby sprzętowe. W każdej sytuacji aplikacja klienta powinna działać płynnie, jeśli tylko sprzęt na którym jest uruchomiona spełnia minimalne wymagania.

Często pliki GIML zawierają więcej niż jedną scenę, które służą do zmiany wyglądu interfejsu użytkownika po zajściu określonych zdarzeń. Każda z kolejnych scen jest generowana analogicznie, jak w przypadku sceny domyślnej. Zasoby z poprzedniej sceny są zwalniane z pamięci wykorzystywanej przez program. Jest to szczególnie istotne podczas konstruowania dużych aplikacji z wieloma rodzajami dostępnych interfejsów użytkownika. Wyjście z aplikacji może być wywołane z poziomu okna konfiguracyjnego lub przy użyciu klawisza ESC.

Ważnym elementem pozostaje sposób sterowania aplikacją. Połączenie z eyetrackerem

oznacza, że ciągle wymieniane są dane pomiędzy komputerem a serwerem eyetrackera. Musi być więc zapewniona stabilność połączenia sieciowego. Proponuje się wykorzystywać do tego celu łącza przewodowego ze względu na jego stabilność, wysoki transfer i odporność na zakłócenia. Sygnał z sieci bezprzewodowych może być czasami utracony a w przypadku korzystania z okulografu ciągłość wymiany danych jest podstawą poprawnego działania. Zgodnie z tym, co zostało opisane wcześniej wymiana ta obywa się poprzez protokół TCP/IP pomiędzy gniazdami[1]. Komunikację tą zapewniają sterowniki eyetrackerów. W przypadku SMI, sterownik jest dostarczany w postaci SDK [2]. Jest to wygodny sposób dla programistów, ponieważ zawiera on gotową implementację, którą należy dodatkowo "opakować", aby umożliwić ich użycie we własnym projekcie. Ponieważ dostarczony SDK jest oparty na standardach firmy SMI, możliwa jest obsługa różnych eyetrackerów tej firmy. W przypadku Mirametrixa sytuacja jest nieco odmienna. Wymaga większego nakładu pracy. Eyetrackery Mirametrix korzystają ze standardu wprowadzony m.in. przez znaną osobę w świecie eyetrackingu - prof. Duchowskiego [18], który zaproponował *Open Eye-gaze Interface* jako otwarty standard sterownika dla eyetrackerów. Okazuje się, że poza Mirametrixem również inne firmy wykorzystują ten standard (m.in. popularna firma Tobi). Głównym jego założeniem jest komunikacja z okulografem za pomocą przesyłanych poleceń w języku XML. Polecenia te mają określoną strukturę: rodzaj polecenia (GET, SET), identyfikator (zawiera nazwę funkcji, np. rozpoczęcie procesu kalibracji) oraz parametry, jeśli w danej funkcji są wymagane. Każde z poleceń wysyłanych z komputera jest potwierdzane przez serwer eyetrackera poleceniem ACK lub NACK. W przypadku kiedy nie dotrze potwierdzenie należy ponownie wysłać żądanie. GCAF posiada oddzielną bibliotekę, która implementuje ten standard w języku C# dzięki czemu może być pierwszym dostawcą SDK dla tego rozwiązania. Obecnie na świecie nie zaproponowano podobnego udogodnienia, co byłoby dużym ułatwieniem dla przyszłych programistów chcących wdrożyć obsługę eyetrackerów w swoich programach.

[1]Gniazdo to zestaw adresu IP oraz portu, po których nawiązywana jest komunikacja z daną aplikacją

[2]SDK (Software Development Kit to zestaw narzędzi przeznaczonych dla programistów celem umożliwienia wykorzystania danej biblioteki.

4.5 Dane zapisywane przez program

Platforma GCAF zapisuje szereg danych do plików. Są to:

1. Jednym z nich jest oczywiście sekwencja zrzutów ekranu z przebiegu badania.

2. Istnieje także możliwość nagrania filmu. O tej możliwości już wcześniej wspomniano.

3. Ważne są również statystyki dotyczące obszarów zainteresowań, które są wybierane jeszcze przed uruchomieniem aplikacji. Zgodnie z tym, co było pokazane w oknie konfiguracyjnym, mogą one zostać zapisane do trzech popularnych formatów plików: CSV, TXT oraz XML. W przypadku formatu CSV oraz TXT statystyki są zapisywane przy użyciu odpowiedniego separatora, co ułatwia ich późniejszy odczyt w programach przeznaczonych do obróbki danych. Jeśli chodzi o plik XML dane te są zapisywane jako zserializowane obiekty statystyk, tworzonych przez GCAF. Pliki XML dają więc możliwość odtworzenia tych obiektów w innych programach, które wykorzystują te statystyki do dalszej obróbki. Wystarczy dokonać procesu deserializacji aby uzyskać z powrotem wypełniony obiekt statystyk. Fragment takiego pliku widoczny jest na rys. 4.12. Parametry, które są zawarte w plikach statystyk dotyczą analizy fiksacji, sakkad oraz mrugnięć (mrugnięcia będą obsługiwane dopiero w kolejnej wersji programu GCAF). Opisują takie parametry, jak ich liczba, czas najdłuższego i najkrótszego trwania, czas średni, amplitudę, prędkość, itp. Wszystkie te dane są istotne z punktu widzenia osób zajmujących się okulografią i wykorzystują je w badaniach.

4. Innymi plikami, które również znajdują się w elementach wynikowych programu są tzw. logi z przebiegu działania aplikacji, zmiany jej stanów, komunikatów otrzymywanych z eyetrackera, zmiany stanu aktywności obszarów oraz scen, a także zmiany pozycji wzroku, czy alerty, które wystąpiły podczas badania. Wszystkie pliki, zawierające wymienione informacje są zapisywane do plików tekstowych pod ścieżkami, które zostały podane przez użytkownika w oknie konfiguracji. Analiza tych plików pomaga w znalezieniu błędów pojawiających się podczas pracy programu, ale przede wszystkim są źródłem informacji o stopniu zainteresowania badanego danymi obiektami i jego reakcji na pojawiające się zdarzenia.

Rys. 4.12 Fragment pliku XML zawierający statystyki obszarów zainteresowania

Plany dotyczące kolejnych wersji platformy GCAF zawierają wdrożenie takich elementów jak mapy cieplne oraz mapy fiksacji. Byłyby one graficzną reprezentacją danych, które są zbierane i zapisywane do plików. Ich zastosowanie mogłoby sprawić, że platforma GCAF znalazła by szersze zastosowanie podczas zabiegów marketingowych - np. przy analizie stron www.

5. Eksperymenty z użyciem GCAF

Możliwości zastosowania platformy GCAF są bardzo szerokie ze względu na grono osób, do których jest ona dedykowana. Ostatni rozdział niniejszej pracy został poświęcony przykładom użycia GCAF w eksperymentach prowadzonych przez Interdyscyplinarne Centrum Nowoczesnych Technologii. Przedstawione badania będą opisane od strony programistycznej tworzenia aplikacji bez zagłębiania się w szczegóły otrzymywanych wyników.

5.1 Bajka interaktywna

Pierwszym badaniem, w którym użyto GCAF była interaktywna bajka dla niemowląt. Głównie z myślą o niej stworzono tę platformę. Pomysł stworzenia interaktywnej bajki był ściśle związany z prowadzonym obecnie w ICNT projektem stworzenia interaktywnej zabawki. Bajka jest tylko jedną z części tego przedsięwzięcia. Głównym jej założeniem było stworzenie aplikacji, która miała trenować u dzieci zdolność rozpoznawania fonemów językowych, co pomogłoby dzieciom szybciej uczyć się obcych języków w późniejszych latach życia. Wykorzystanie do tego celu eyetrackera było przemyślanym rozwiązaniem, gdyż jest to jedyny sposób, w jaki małe dzieci mogą sterować komputerem.

Stworzenie odpowiedniego scenariusza wymagało współpracy osób z różnych dziedzin. Istotny był również wkład grafika - Łukasza Goraczewskiego, który musiał tworzyć odpowiednie rysunki, aby zainteresować tym małych odbiorców. Ważnym zadaniem było stworzenie scenariusza bajki i jego implementacja w pliku GIML. Był to pierwszy, tak zaawansowany program z użyciem tej technologii. Przykładowe rezultaty jakie uzyskano przedstawia rys. 5.1.

Sam pomysł i dobrze przemyślana realizacja spowodowały, że eksperymentem zainteresowała się telewizja, która przedstawiła bajkę w reportażu.

Rys. 5.1 Wygląd interfejsu użytkownika dla bajki interaktywnej stworzonego przy użyciu języka GIML

5.2 Czerwona kropka

Funkcjonalność GCAF została także doceniona przez osoby tworzące eksperyment o nazwie *czerwona kropka*. Polegał on na umieszczeniu na lewej stronie ekranu obszaru z filmem, a z prawej strony - obszaru przedstawiającego czerwoną kropkę (rys. 5.2. Aplikacja działała w ten sposób, że po jej uruchomieniu odtwarzany był film. Wywołanie zdarzenia reakcji na kropce powodowało przełączanie się do kolejnej sceny, która wyglądała analogicznie jak poprzednia, z tą różnicą, że zmieniany był film (czyli kropka służyła do przełączania filmów). Badanie polegało na analizie, po jakim czasie dziecko zorientuje się, do czego służy umieszczona na ekranie czerwona kropka. W odróżnieniu od poprzedniego badania, wygląd aplikacji był bardzo uproszczony i nie wymagał współpracy z grafikami. Jedyną trudnością było dopasowanie odpowiednich filmów.

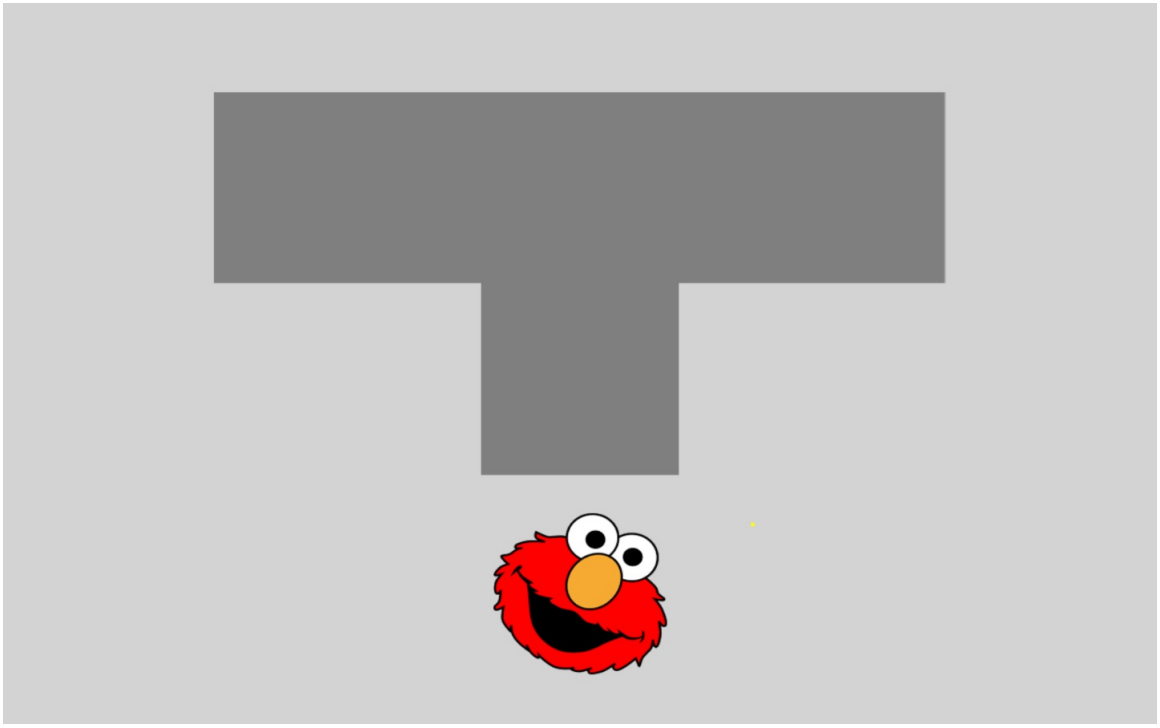
Eksperyment obecnie przeprowadzany jest z filmami, które są fragmentami popularnych bajek. Aktualnie są wpierane takie języki jak polski, angielski, niemiecki, francuski, węgierski i mandaryński.

Rys. 5.2 Wygląd interfejsu użytkownika dla eksperymentu *czerwona kropka* z użyciem języka GIML

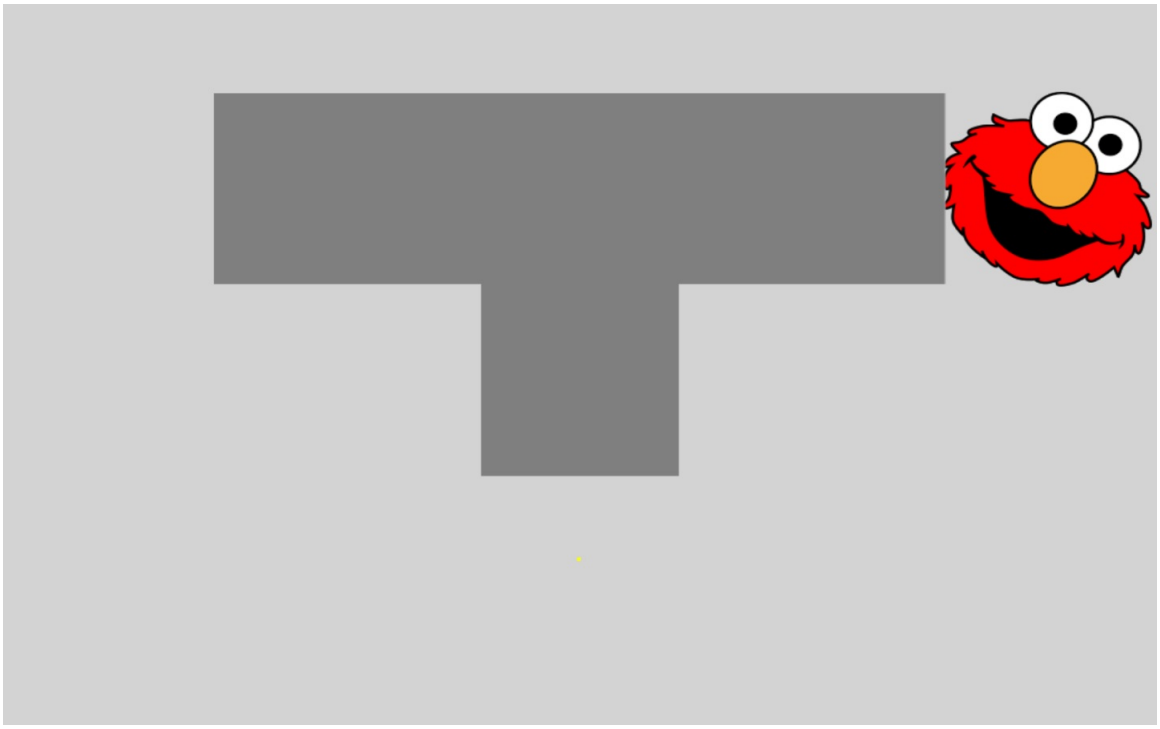
5.3 Elmo

Elmo to nazwa wewnętrzna kolejnego eksperymentu realizowanego w ICNT przy użyciu GCAF. Dotyczy problemu rozpoznawania fonemów u niemowląt oraz czasu, po którym umiejętność ta zostaje zatracana. Aplikacja dla tego eksperymentu przedstawia obszar, zawierający znaną postać z animowanych bajek, która ma przyciągnąć uwagę dziecka, usytuowanego centralnie u dołu ekranu. Na środku sceny znajduje się przesłona w kształcie litery T, który został uzyskany poprzez złożenie dwóch prostokątnych obszarów. Kiedy wzrok dziecka przesunie się na obszar z postacią uruchamiany jest dźwięk charakterystycznych podobnych słów w różnych językach. Sam obiekt natomiast zaczyna poruszać się ku górze w stronę tunelu. Kiedy do niego dotrze chowa się pod przesłoną i pojawia się z jednej, bądź drugiej strony. Schemat ten dobrze ilustrują rysunki 5.3 oraz 5.4.

Celem badania jest sprawdzenie, czy dziecko jest w stanie rozróżnić aktywowany dźwięk i spojrzeć w odpowiednie wyjście tunelu (po prawej lub po lewej stronie) kiedy obiekt zostanie chwilowo w nim ukryty. Dźwięki, które są odtwarzane, są bardzo podobnie brzmiące (różnią się wymową poszczególnych samogłosek). Jeśli dziecko zostane poddane



Rys. 5.3 Wygląd interfejsu użytkownika dla eksperymentu *elmo* przed aktywacją obszaru



Rys. 5.4 Wygląd interfejsu użytkownika dla eksperymentu *elmo* po aktywacji obszaru

badaniu kilkukrotnie, powinno odpowiednio reagować na dane słowo kierując wzrok w odpowiednią stronę. W oczekiwanych miejscach pojawienia się postaci znajdują się ukryte obszary, które zbierają informację o uwadze dziecka.

5.4 Koła

Ostatnim z eksperymentów aktualnie prowadzonym w ICNT są tzw. "koła". Scena zostaje wypełniona obszarami o kształcie dwóch kół, o tym samym środku. W obszar mniejszego koła zostaje załadowany rysunek przedstawiający zwierzątko. Koła zewnętrzne są wypełnione kolorową teksturą. Na jednej scenie znajdują się trzy różne koła (rys. 5.5). Każde z nich, po uruchomieniu aplikacji wykonuje inną animację - obrót, powiększanie, itp.

Rys. 5.5 Wygląd interfejsu użytkownika dla eksperymentu *kola* dla jednej z wielu dostępnych scen

Kiedy wzrok znajdzie się na którymś z kół wówczas uruchamiana jest animacja wewnętrznego obszaru wraz z dźwiękiem narratora, który przedstawia się jako dana postać i zachęca dziecko do wspólnej zabawy. Przy aktywacji danego koła pozostałe obiekty znajdujące się na scenie zostają przyciemnione aby nie rozpraszać uwagi dziecka. Na scenie znajduje się zawsze jeden obszar, którego aktywowanie prowadzi do zmiany sceny. Taki przykład zastosowania GCAF może również posłużyć jako interaktywna bajka dla dzieci. Scena jest zmieniana po aktywacji wszystkich obszarów.

6. Podsumowanie

Możliwości stosowania okulografów w życiu codziennym są bardzo szerokie. Ciągle odkrywane są jednak nowe. Niewątpliwą zaletą eyetrackerów jest możliwość ich używania przez osoby, mające problem z posługiwaniem się powszechnymi urządzeniami sterującymi, ale również pozwalają uzyskać cenne informacje. W przypadku platformy GCAF pozyskanie tych informacji jest dużo łatwiejsze, dzięki analizie, która jest dokonywana już w trakcie działania aplikacji. GCAF oszczędza cenny czas badaczom oraz co najważniejsze obniża koszty badań.

GCAF jest przykładem produktu, który łączy w sobie dwa nowatorskie podejścia - sterowanie wzrokiem oraz możliwość tworzenia własnych aplikacji. Możliwości sprawiają, że GCAF jest tylko podstawą tych programów, które mogą zostać za jej pomocą stworzone. Pomimo dużej złożoności i funkcjonalności platformy, jest ona dla użytkownika przejrzysta, a sposób konfiguracji starannie uporządkowany i czytelny. Prowadzone testy oraz dotychczasowe użycie platformy pokazuje, że nie występuje problem z jej obsługą przez osoby niedoświadczone, nawet w przypadku tworzenia własnego interfejsu aplikacji przy użyciu języka GIML.

Przyszłość platformy zależy w głównej mierze od sukcesów badań, w których jest ona wykorzystywana. Jednak jej główną zaletą jest niezależność działania oraz dopasowanie do wymagań klientów. Zastosowanie języka GIML jest z pewnością podejściem nowym, który może być początkiem ogólnie przyjętego standardu tworzenia aplikacji. Doświadczenie związane z prezentacją platformy GCAF oraz jej zalet na różnych wystąpieniach, zarówno w świecie specjalistów branży IT, jak i eyetrackingu, pozwala wnioskować, że produkt ten posiada ogromny potencjał i duże szanse zaistnienia na rynku.

7. Bibliografia

[1] Holmqvist K. Nyström M. Andersson R., Dewhurst R., Jarodzka H., Weijer J. v. d., *Eye Tracking. A Comprehensive Guide to Methods and Measures*, New York, Oxford University Press, 2011

[2] Daunys G. *Gaze Interaction and Applications of Eye Tracking: Advances in Assistive Technologies (Chapter 22 - Eye Tracker Hardware Design)*, Hershey, IGI Global, 2012

[3] Mulvey F. *Gaze Interaction and Applications of Eye Tracking: Advances in Assistive Technologies (Chapter 2 - Eye Anatomy, Eye Movements and Vision)*, Hershey, IGI Global, 2012

[4] Duchowski A. *Eye Tracking. Methodology. Theory and Practice.*, Clemson, Springer-Verlag, London, 2007

[5] Michalski R., Grobelny J., Jach K., Kuliński M., *Wykorzystanie okulografii w analizie użyteczności serwisów internetowych*, Wrocław, Instytut Organizacji i Zarządzania, Politechnika Wrocławska, Warszawa, 2006

[6] Jaworski A., Kuszewski H., Lejda K., Ustrzycki A., Woś P. *Okulograf jako narzędzie wspomagające audyt BRD*, Visnik Nacionalnogo Transportnogo Universitetu, t.27, Kijów, 2013

[7] Żydzik K., Rak T. *C# 6.0 i MVC5. Tworzenie nowoczesnych portali internetowych*, Helion, Gliwice, 2015

[8] Thite L., Brown R. *The history of Eye tracking*, South Africa, Department of Computer Science and Informatics, Uniwersity of the Free State, South Africa

[9] Strona internetowa dot. eyetrackingu: http://eyetracking.pl/

[10] Strona internetowa dot. zagadnień optycznych: http://optyczne.pl/

[11] Strona internetowa dot. zagadnień optycznych: https://www.teledynedalsa.com/imaging/knowledge-center/appnotes/ccd-vs-cmos/

[12] Strona internetowa dot. tworzenia serwisów internetowych:
http://www.cms.rk.edu.pl/w/p/sledzenie-kliniec-i-mapy-cieplne/

[13] Strona internetowa portalu SEOPortal:
http://www.seoportal.pl/eye-tracking-w-sluzbie-seo-i-sem-czyli-gdzie-wlasciwie-patrza-internauci

[14] Strona internetowa MSDN:
https://msdn.microsoft.com/pl-pl/library/

[15] Strona internetowa wikipedii o języku XML:
https://pl.wikipedia.org/wiki/XML

[16] Strona internetowa dot. eyetrackingu:
http://www.eyetracker.pl/oferta-view/obszary-badan/

[17] Strona internetowa Kamila Durkiwicza:
http://durkiewicz.blogspot.com/2012/01/programowanie-deklaratywne-vs.html

[18] Strona internetowa Andrewa T. Duchowskiego:
http://andrewd.ces.clemson.edu/