AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa magisterska

Opracowanie prototypu interfejsu dla gier wykorzystujących pętlę afektywną

Development of a prototype interface for games based on the affective loop

Autor:	*Kamil Osuch*
Kierunek studiów:	*Informatyka*
Opiekun pracy:	*prof. dr hab. inż. Grzegorz Jacek Nalepa*

Kraków, 2015

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

1. Wstęp

W ciągu ostatnich lat pojęcie sztucznej inteligencji przestało być tylko fenomenem, który dla większości społeczeństwa istniał pod postacią filmów lub powieści dotyczących maszyn posiadających świadomość. Od tamtego momentu człowiek zaczął znajdować zastosowanie sztucznej inteligencji w coraz większej liczbie dziedzin. Od maszyn przetwarzających w sposób automatyczny ogromne ilości informacji, aż po systemy gromadzące dane dotyczące użytkowników i na ich podstawie generują reguły, dzięki którym możliwe jest znalezienie rozwiązań i porad dopasowanych do danego użytkownika.

Co jest w tym wszystkim najistotniejsze, to fakt, że człowiek otacza się sztuczną inteligencją, nawet tego nie zauważając. Systemy rekomendujące, dopasowujące produkty z każdej tematyki do preferencji użytkowników [1], urządzenia nasobne monitorujące nasz stan zdrowia dzięki pomiarom parametrów życiowych i zbieraniu informacji na temat naszych nawyków [2], czy coraz popularniejsze systemy autonomicznej jazdy [3]. Sztuczna inteligencja z dnia na dzień coraz bardziej wnika w każdy aspekt życia człowieka.

Jednym z elementów, który odgrywa ważną rolę w życiu człowieka, a który coraz mocniej oparty jest o sztuczną inteligencję, są gry komputerowe. Choć początkowo były one traktowane wyłącznie jako rozrywka, to dziś coraz częściej są określane nawet jako pewna forma nauki konkretnych umiejętności [4]. Przykładem mogą być tutaj gatunek gier strategicznych, które uczą taktyki i zarządzania, a także gry logiczne pozwalające na rozwój logicznego myślenia. Warto tutaj wspomnieć także o grach poważnych [5], które nie skupiają się na aspektach rozrywkowych, a bardziej są formą edukacji i szkoleń przedstawionych w formie interaktywnej symulacji.

Pojęciem, które jest coraz szerzej widoczne w kontekście sztucznej inteligencji związanej przede wszystkim z tematyką komunikacji człowiek-komputer jest informatyka afektywna. Chociaż samo pojęcie istnieje już od ponad 20 lat [6], to dopiero w ciągu kilku ostatnich badania w tej tematyce stały się popularne [7]. Początkowo główną ideą tej dziedziny były systemy zbierające i analizujące dane na temat stanów emocjonalnych użytkowników. Ponieważ emocje nie mogą być kontrolowane poprzez działania człowieka, a jednocześnie można je opisać między innymi przy pomocy zmian fizjologicznych w ludzkim ciele, wykorzystanie informatyki afektywnej w systemach inteligentnych takich jak aplikacje rekomendujące czy systemy badające stan zdrowotny użytkowników pozwala na zwiększenie ich skuteczności działania.

W podobny sposób powstała próba powiązania dziedziny informatyki afektywnej z grami komputerowymi, tworząc nowy rodzaj gier, nazywanych grami afektywnymi. Główną ich ideą jest pomiar stanów

emocjonalnych wywoływanych na użytkowniku w trakcie rozgrywki oraz dostosowywanie gry w czasie rzeczywistym do odczytanych reakcji gracza, tak aby zwiększyć doznania płynące z gry [8]. Dzięki temu każda gra może zostać w pewien sposób spersonalizowana na podstawie indywidualnych cech użytkownika.

Celem pracy jest opracowanie prototypu dwuczęściowego interfejsu umożliwiającego pomiar sygnałów pozwalających na określenie zmian stanów emocjonalnych gracza. Interfejs ma posłużyć do opracowania prototypów gier zawierających pętlę afektywną. W skład interfejsu będą wchodzić:

– zbiór urządzeń umożliwiających pomiary sygnałów wykorzystanych do określenia zmian emocji gracza

– moduł przygotowany w środowisku do tworzenia gier, który na podstawie zgromadzonych z urządzeń pomiarowych sygnałów będzie określał stan emocjonalny i zachowania użytkownika

Ważnym elementem pracy jest stworzenie gry zawierającej pętlę afektywną [9], która będzie wykorzystywała przygotowany interfejs. Po wykryciu zachowań oraz zmian stanów emocjonalnych użytkownika, stan gry jest aktualizowany.

Niniejsza praca składa się z 8 rozdziałów. W rozdziale 2 zostały przedstawione podstawy teoretyczne dotyczące informatyki afektywnej. Szczególny nacisk położono na tematykę gier afektywnych, przedstawiając wybrane istniejące rozwiązania, problemy i kierunki badań z tego zakresu. Rozdział 3 zawiera przedstawienie oraz analizę dostępnych sprzętowych platform pomiarowych, możliwych mechanizmów wnioskowania oraz narzędzi wykorzystywanych do budowy gier komputerowych. Ważnym elementem jest przedstawienie wad i zalet każdego z rozwiązań w kontekście tematyki pracy. Rozdział 4 jest podsumowaniem analizy platform sprzętowych z poprzedniego rozdziału. Przedstawione tu zostały podstawowe założenia, jakie powinny być spełnione przez wybraną grupę urządzeń pomiarowych, a także definiuje sprzęt wybrany podczas końcowej implementacji. W rozdziale 5 opisany został proces budowy modelu do rozpoznawania emocji. Omówione zostały wybrane zbiory danych, sposób ich przetwarzania, oraz budowa i wybór końcowego modelu na podstawie działania z dostępnymi danymi. Rozdział 6 jest jedną z najistotniejszych części pracy. Przedstawiono w nim proces implementacji utworzonej gry komputerowej z pętlą afektywną. Skupiono się na opisie interfejsu łączącego rozwiązania wybrane w poprzednich rozdziałach i sposobie jego wykorzystania wewnątrz gry. Następnie przedstawione zostały mechaniki pokazujące, w jaki sposób przygotowany moduł może wpłynąć na rozgrywkę tak, by sprzężenie zwrotne mogło zostać zamknięte. W rozdziale 7 opisany został sposób ewaluacji stworzonego rozwiązania oraz charakterystyka i proces przeprowadzonych eksperymentów. Ostatni rozdział stanowi podsumowanie niniejszej pracy. Zawiera wnioski dotyczące przygotowanego projektu, jego mocne i słabe strony. Opisane zostały także możliwe kierunki dalszego rozwoju projektu.

2. Informatyka afektywna

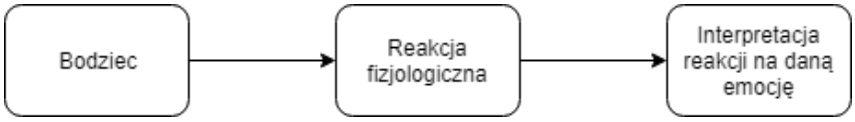
Informatyka afektywna (ang. *affective computing*) jest dziedziną informatyki, w której obliczenia są powiązane z emocjami, lub bezpośrednio na nie wpływają [6]. Głównym celem informatyki afektywnej jest rozpoznawanie oraz analiza emocji ludzkich, możliwość ich symulacji przez komputer, a także wpływanie na emocje użytkownika poprzez konkretne bodźce. Aby uzyskać taki efekt, informatyka afektywna jest silnie powiązana z dziedzinami takimi jak psychologia, fizjologia, czy kognitywistyka [10].

2.1. Emocje

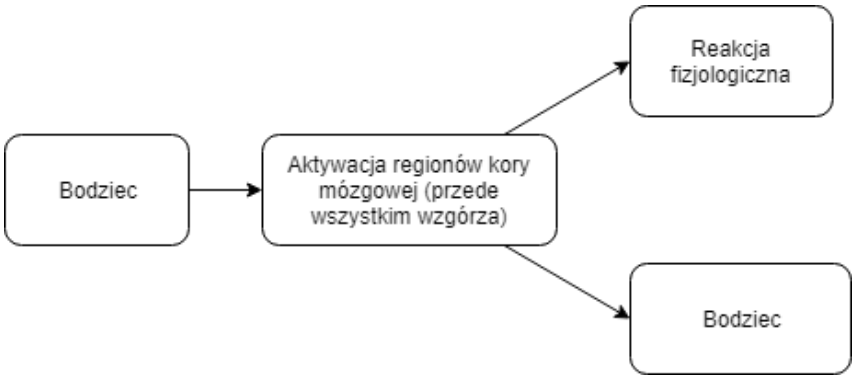
Choć do dzisiaj nie ma jednej, uniwersalnej definicji czym są emocje, to środowisko naukowe ciągle przeprowadza badania na tematy powiązane z emocjami. Już w XIX wieku powstała teoria opracowana niezależnie przez Williama Jamesa oraz Carla Lange'a, w której emocję zdefiniowano jako interpretację reakcji cielesnej na zaobserwowany bodziec [11]. Obaj badacze przyjęli, że na konkretny czynnik człowiek reaguje najpierw reakcją fizjologiczną, która następnie zostaje przez niego przypisana do wzorca odpowiadającego danej emocji. Dla przykładu, jeśli człowiek znajdzie się w sytuacji, w której zobaczy zagrożenie, zaczyna się trząść i pocić, a jego tętno gwałtownie wzrasta, jego mózg natomiast interpretuje to jako strach.

Teoria Jamesa-Lange'a została zakwestionowana w latach dwudziestych XX wieku przez dwójkę naukowców, Waltera Cannona i Phillipa Barda. Zasugerowali oni, że odczuwanie emocji nie jest zależne od reakcji fizjologicznych, a raczej są to reakcje zachodzące jednocześnie jako odpowiedź na dany bodziec [12]. Teoria ta była bezpośrednim zakwestionowaniem badań przeprowadzonych przez Jamesa i Lange'a. Jej autorzy przeprowadzili badania na kotach, na podstawie których przedstawili, że to wzgórze jest obszarem mózgu odpowiedzialnym za reakcje emocjonalne na doświadczane bodźce. Cannon zauważył, że całkowite odcięcie wszystkich układów od mózgu nie zmienia zachowania emocjonalnego zwierząt, co kłóciło się z teorią Jamesa-Lange'a, według której koty powinny przestać wykazywać jakiekolwiek reakcje emocjonalne.

Teoria Jamesa-Lange'a została ponownie podjęta przez Prinza, który przedstawił więcej dowodów na potwierdzenie tej teorii, jednocześnie opisując argumenty pokazujące, że reakcje fizjologiczne nie zawsze są wystarczające, lub w ogóle niepotrzebne do odczuwania emocji [13]. Zwrócił on między innymi uwagę na badania Hohmanna nad pacjentami z urazami rdzenia kręgowego [14], w których zauważono



(a) Teoria Jamesa-Lange'a, źródło: opracowanie własne na podstawie [11]



(b) Teoria Canona-Barda, źródło: opracowanie własne na podstawie [12]

Rys. 2.1. Graficzne porównanie teorii emocji Jamesa-Lange'a i Canona-Barda

efekty zarówno potwierdzające, jak i podające w wątpliwość teorię Jamesa-Lange'a. Hohmann zauważył, że u osób z urazem można zauważyć redukcję w odczuwaniu emocji, ale niektóre z nich wciąż były zauważalne. Prinz komentuje to jako zdolność mózgu do przewidzenia reakcji fizjologicznej na dany bodziec, co z kolei będzie skutkować odczuwaniem emocji. Jest to możliwe dzięki wcześniejszym doświadczeniom. Tak jak człowiek, który stracił wzrok, jest w stanie wyobrazić sobie przedmiot, który kiedyś widział, tak samo mózg potrafi określić jaka reakcja fizjologiczna mogła nastąpić na dany bodziec.

2.2. Modele emocji

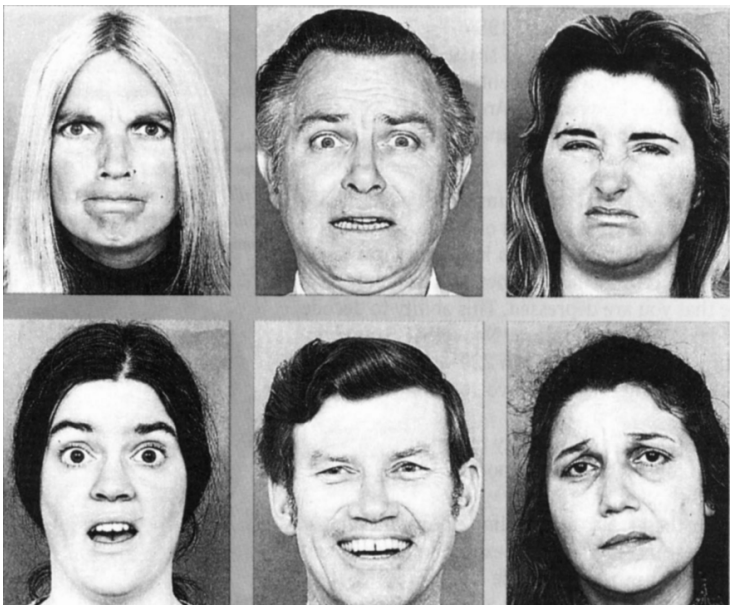
Ponieważ emocje są pojęciem abstrakcyjnym, ich pomiar oraz analiza w informatyce afektywnej nie jest prosta i wymaga przyjętego modelu, który pozwoli na jednoznaczny pomiar emocji. Na przestrzeni lat powstało wiele modeli opisujących emocje, które można podzielić na dwie główne kategorie [15]:

– modele kategoryzowane, w których określony jest zbiór emocji reprezentowanych przez konkretne oznaczenia

– modele wymiarowe, w których emocje są reprezentowane przy pomocy zbioru miar określających ich własności.

2.2.1. Modele kategoryzowane

Do typu modeli kategoryzowanych zaliczają się przede wszystkim modele emocji bazowych. Na przestrzeni lat powstało wiele modeli różniących się od siebie ilością oraz rodzajami emocji. Przykładem może być tutaj model zaproponowany przez Oatley'a i Johnsona-Lairda, w którym proponują oni pięć podstawowych emocji: gniew, lęk, odraza, radość i smutek [16]. Jednym z popularniejszych modeli emocji bazowych, jest ten przedstawiony przez Ekmana. Razem z Friesenem przeprowadzili badania na plemieniu z Papui Nowej Gwinei. Członkowie plemienia byli w stanie zidentyfikować sześć emocji: strach, gniew, wstręt, smutek, szczęście i zaskoczenie (rys. 2.2) [17]. Kilka lat później do tej grupy Ekman dodał pogardę, aby odróżnić ją od wstrętu.

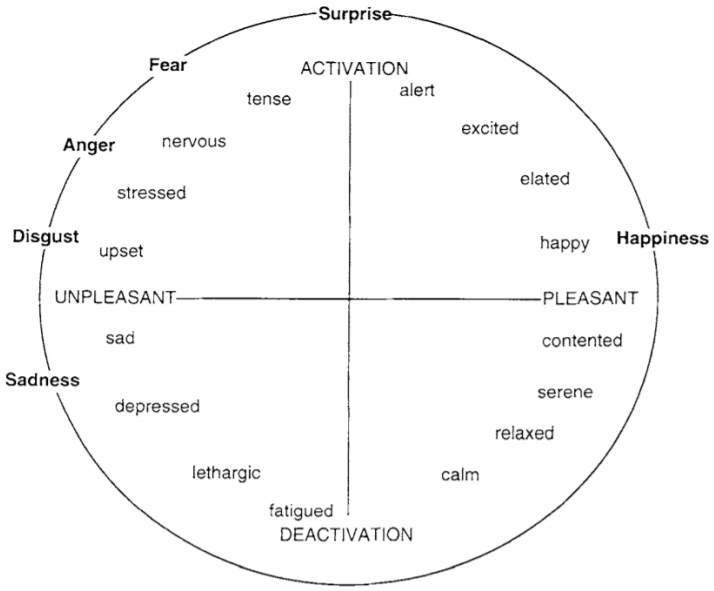


Rys. 2.2. Wyrazy twarzy odpowiadające sześciu emocjom zaproponowanym przez Ekmana, źródło: [17]

W modelach kategoryzowanych często nazywanych także modelami dyskretnymi emocje są niezależne od siebie. Ich dużą zaletą jest to, że automatycznie reprezentują ludzkie emocje dzięki łatwym do zrozumienia etykietom. Jednak w swojej pracy James Russell i Lisa Barret przedstawiają problemy, jakie można napotkać w tym typie modeli [18]. Pierwszym są różnice występujące w nazwach kategorii w zależności od języka. Każdy ze stanów emocjonalnych może być opisany na różne sposoby w zależności od języka. Różnice mogą wystąpić nie tylko w samym tłumaczeniu, ale nawet w ilości słów opisujących daną emocję. Drugim problemem jest to, że granice pomiędzy danymi kategoriami emocji często są rozmyte. Te same stany emocjonalne można wyrazić za pomocą różnych kategorii w zależności od różnic kulturowych, środowiskowych czy osobowościowych [15]. Trzecim przedstawionym problemem jest to, że każda z kategorii emocji, takich jak gniew, czy strach składają się z zestawu uporządkowanych w czasie, powiązanych ze sobą zdarzeń. W związku z tym takie emocje powinno traktować jako złożone procesy, które nie powinny być traktowane jako elementy atomowe.

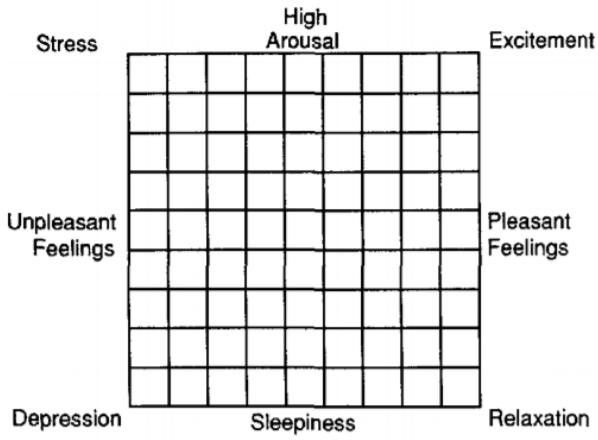
2.2.2. Modele wymiarowe

Russell w swoich badaniach zauważył jeszcze jeden istotny problem modeli dyskretnych. Ponieważ opisują one stany emocjonalne w sposób jednoznaczny, nie można określić poziomu odczuwania emocji. Jako przykład podaje porównanie strachu w trakcie przejażdżki kolejką górską a sytuacjami zagrażającymi życiu. W przypadku modeli dyskretnych w obu sytuacjach można jedynie określić odczuwanie emocji strachu, pomimo różniących się reakcji fizjologicznych. W ramach swoich badań Russel zaproponował model w kształcie okręgu, składający się z dwóch wymiarów (rys. 2.3). Pierwszym z nich jest poziom odczuwania przyjemności w modelu nazwany wartościowością (ang. *valence*). Odzwierciedlał on także pozytywność lub negatywność emocji. Niskie wartości odpowiadają emocjom takim jak smutek, stres, czy odraza, natomiast szczęście lub ekscytacja są opisane przez wysokie wartości wartościowości. Drugim wymiarem jest pobudzenie (ang. *arousal*), które opisuje natężenie danej emocji. Odczucia takiej jak senność, depresja czy zrelaksowanie charakteryzują się niskimi wartościami pobudzenia, natomiast wartościom wysokim odpowiadają emocje takie jak ekscytacja, zaskoczenie, czy złość. Model został przedstawiony w formie koła, na którego brzegach opisane zostały podstawowe emocje. Warto wspomnieć, że nie jest to pierwszy model o strukturze kołowej. Sam Russel, chcąc pokazać uniwersalność swojego modelu, porównuje go między innymi z teoriami zaproponowanymi przez Watsona i Tellegena [19]. Tym, co wyróżniało model Russela, były przeprowadzone badania [20]. Russel wybrał 28 słów reprezentujących konkretne stany afektywne i przy pomocy trzech różnych technik przeskalował je, otrzymując zbliżone wyniki. Aby potwierdzić swoją teorię, przeprowadził eksperyment z grupą 36 osób, która miała przyporządkować wybrane słowa do ośmiu kategorii określających podstawowe stany emocjonalne, a następnie ułożyć te kategorie na modelu kołowym w taki sposób, aby przeciwne emocje znalazły się po przeciwnych stronach koła.



Rys. 2.3. Model wymiarowy zaproponowany przez Russela, źródło: [18]

Kilka lat później Russel wraz z Anną Weiss oraz Geraldem Mendelsohnem przedstawił propozycję jednopunktowej skali nazwaną Affect Grid, która miała służyć jako sposób szybkiej i prostej do analizy oceny stanów afektywnych wzdłuż wymiarów znanych z modelu Russela, wartościowości oraz pobudzenia [21]. Skala jest przedstawiona w formie siatki, gdzie oba wymiary określane są w zakresach od 1 do 9 w sposób ciągły. Początkowo miała ona mieć formę kołową, podobną kołowej struktury modeli emocji, jednak ostatecznie zrezygnowano z niej na rzecz siatki, która była znacznie prostsza do wyjaśnienia badanym osobom. W ramach sprawdzenia, czy przedstawiona skala w poprawny sposób odzwierciedla stany emocjonalne, przeprowadzono badania, w których wykorzystano między innymi słowa wybrane przy wcześniejszych badaniach Russela oraz zbiór zdjęć z wyrazami twarzy odpowiadającymi konkretnym stanom afektywnym. Zadaniem badanych było oznaczenie na skali poziomu wartościowości i pobudzenia, które najlepiej odzwierciedlały dane słowo lub wyraz twarzy. Następnie wyniki porównano z innymi skalami opisującymi poziomy wartościowości i zauważono wysoki procent podobieństwa między nimi.

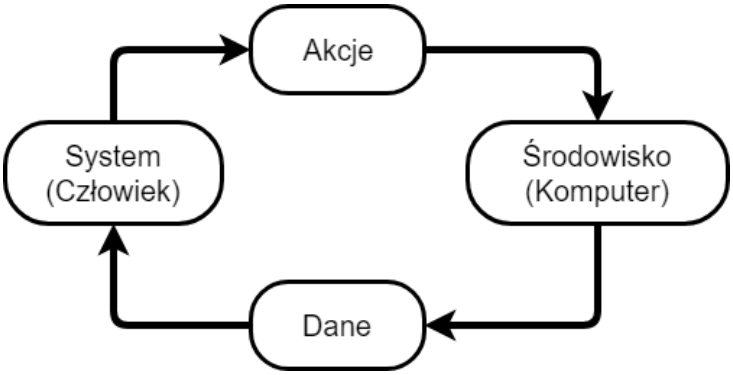


Rys. 2.4. Affect Grid, źródło: [21]

Model emocji przedstawiony przez Russela oraz skala opisana w Affect Grid stały się jednymi z bardziej popularnych form przedstawiania emocji w informatyce afektywnej. Ogromną zaletą jest przede wszystkim łatwość przetwarzania danych opartych na dwuwymiarowej skali, które określają dane stany emocjonalne. Skalę tą można zauważyć między innymi w zbiorach danych zawierających powiązania między reakcjami fizjologicznymi a odpowiednimi wartościami wartościowości i pobudzenia [22, 23].

2.3. Pętla afektywna

Do realizacji celów zakładanych przez informatykę afektywną [6] wykorzystywana jest pętla afektywna. To pochodzące z dziedziny Automatyki i Robotyki pojęcie zakłada, że system oraz środowisko są zaangażowane w ciągłą interakcję między sobą. Schemat pętli został przedstawiony na rysunku 2.5. Kristina Höök przedstawiła definicję pętli afektywnej w formie trzech kroków [9]:



Rys. 2.5. Cykl pętli afektywnej, źródło: opracowanie własne na podstawie [9]

1. Użytkownicy wyrażają emocje poprzez pewne interakcje obejmujące ich ciało, na przykład gesty czy inne akcje wpływające na środowisko.
2. Środowisko następnie odpowiada, generując elementy afektywne, używając na przykład kolorów, animacji czy innej formy, która może wpłynąć na użytkowników.
3. Elementy afektywne wpływają na reakcję użytkowniów, co prowadzi do coraz większego ich zaangażowania w interakcję ze środowiskiem.

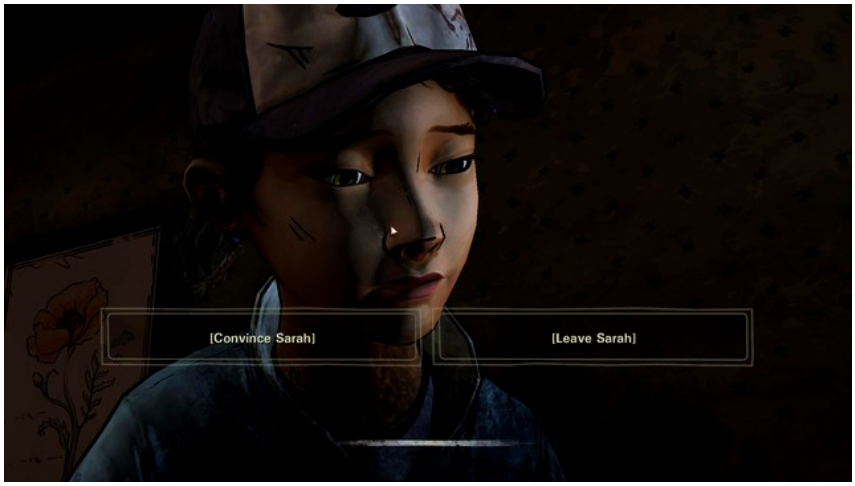
Jak widać, istotnym elementem pętli jest sposób określenia stanów emocjonalnych danej osoby. Spoglądając na rozwiązania praktyczne, można zauważyć, że duże znaczenie ma tutaj wykorzystanie sztucznej inteligencji do przewidywania emocji na podstawie reakcji fizjologicznych [24]. Wykorzystywane są tutaj przede wszystkim wcześniej opisane modele Ekmana [17] oraz Russela [20], które przedstawiają emocję w formie zrozumiałej dla komputera. Same emocje są natomiast przewidywane na podstawie odczytów reakcji fizjologicznych użytkowników, a także ich interakcji ze środowiskiem.

Określenie stanu emocjonalnego użytkownika oraz wzięcie pod uwagę jego interakcji ze środowiskiem pozwala na generację w środowisku sytuacji, które wpłyną na użytkownika tak, aby ten mógł zareagować w odpowiedni sposób i zwiększać swoje zaangażowanie w interakcję ze środowiskiem.

Dobrym przykładem systemów wykorzystujących pętlę afektywną są systemy świadome kontekstu takie jak afektywne odtwarzacze muzyki [25, 26], które na podstawie sygnałów odczytanych od użytkownika modyfikują listę piosenek do odtworzenia, w taki sposób, aby dopasować ją do aktualnego stanu emocjonalnego użytkownika i w jakiś sposób na niego wpłynąć.

2.4. Gry z pętlą afektywną

W ciągu ostatnich lat znacznie wzrosło zainteresowanie grami komputerowymi zawierającymi elementy informatyki afektywnej. Głównym celem tworzenia gier jest dostarczenie użytkownikowi rozrywki poprzez zwiększanie poziomu immersji [27], często również dzięki zapewnieniu wrażeń emocjonalnych. Końcowy odbiór zależny jest między innymi od takich elementów jak fabuła, mechaniki wykorzystane w trakcie rozgrywki, czy też szczegóły wizualne. Aby jak najlepiej zadowolić użytkownika,



Rys. 2.6. Przykład kluczowej decyzji w grze The Walking Dead, źródło: [30]

twórcy gier muszą także wykorzystywać najnowsze technologie, które mogą sprawić, że gracz będzie w stanie jeszcze mocniej odczuwać wrażenia płynące z rozgrywki.

Jednym z takich rozwiązań jest właśnie informatyka afektywna i wykorzystanie emocji do zwiększenia możliwości wpływania na grę. Eva Hudlicka w swoich pracach [28] wskazuje na kluczowe znaczenie emocji w projektowaniu gry. Wywołanie ich u gracza jest możliwe dzięki kluczowym momentom w fabule gry, interakcję z postaciami, do których użytkownik może się w pewien sposób odnieść emocjonalnie, czy nawet poprzez same mechaniki zaimplementowane w grze. Ten sam efekt można także uzyskać przez wygląd środowiska, w którym zostaje umieszczony gracz. Odpowiednie nacechowanie wykorzystanych kolorów, dźwięki otoczenia, czy budowa miejsca, w którym jest postać prowadzona przez gracza, mogą wpłynąć na to, jakie emocje odczuwa gracz. Idealnym przykładem są tutaj modyfikacje graficzne takich tytułów jak Minecraft czy The Elder Scrolls V: Skyrim. Zapewniają one tekstury wyższej jakości, zbliżone jak najbardziej do tego, co gracz może zobaczyć za oknem w realnym świecie. Pozwala to na zwiększeniu immersji poprzez dostarczanie graczowi jak najrealniejszych doświadczeń.

Taki sposób wpływania na emocje Hudlicka nazywa otwartą pętlą. W tym podejściu nie jest wymagane badanie reakcji gracza. Głównym celem jest wpłynięcie na emocje przy pomocy elementów świata gry. Istnieje wiele takich rozwiązań komercyjnych, które, choć wprost nie są nazywane grami afektywnymi, realizują opisane wyżej założenia. Doskonałym przykładem są tutaj gry przygodowe od studia Telltale Games, w których rozgrywka jest formą interaktywnego filmu. W trakcie rozgrywki gracz podejmuje kluczowe decyzje (rys. 2.6), które wpływają na to, jak potoczy się dalsza fabuła gry. Taka forma przedstawienia historii pozwala na zaangażowanie gracza w poznanie postaci i przywiązanie się do nich [29]. Dzięki temu, w kluczowych momentach, takich jak na przykład odejście czy śmierć bohatera, w graczu mogą zostać wywołane adekwatne do wydarzenia emocje.

Drugim sposobem jest wykorzystanie odczuwanych przez gracza emocji do modyfikacji systemów zaimplementowanych w grze tak, aby dostosować je do potrzeb użytkownika i utrzymać jego zaangażowanie. Przykładem zamkniętej pętli, jak nazywany jest ta metoda przez Hudlicką, jest zmiana poziomu

trudności na podstawie samopoczucia gracza. Jeżeli jest on zdenerwowany czy przestraszony, poziom trudności zmniejsza się, natomiast gdy zaczyna on odczuwać nudę, stawiane są przed nim nowe wyzwania, które wzbudzą jego zainteresowanie grą. Zamknięta pętla może być także wykorzystywana w kontekście gier poważnych. Dla przykładu gry wykorzystywane w terapiach mogą na bieżąco monitorować stan emocjonalny użytkownika i modyfikować rozgrywkę w taki sposób, aby osiągnąć konkretny rodzaj emocji.

Ważnym aspektem, na który należy zwrócić uwagę, jest odpowiednie dobranie elementów sprzętowych wykorzystanych do określenia stanu emocjonalnego użytkownika. Ponieważ środowisko odbiorców gier komputerowych wykracza daleko poza aspekty naukowe, istotne jest to, aby urządzenia nie przeszkadzały w rozgrywce, a jednocześnie pozwalały na dokładne określenie, jakie emocje odczuwa gracz. W kolejnych rozdziałach niniejszej pracy omówione zostały przykłady takich urządzeń, wraz z ich zaletami i wadami w kontekście odczytywania emocji, oraz przystępności dla graczy poza warunkami laboratoryjnymi.

Choć świadome wykorzystanie gier afektywnych jest bardziej widoczne w środowisku naukowym i akademickim [31, 32], to producenci gier starają się powoli wprowadzać kontekst emocjonalny w komercyjnych rozwiązaniach jako dodatkowy element urozmaicający rozgrywkę [33, 34]. Jednym z głównych problemów jest niewielka ilość opracowań, w których omówione by zostały dostępne rozwiązania sprzętowe umożliwiające predykcję emocji, oraz to, w jaki sposób zaimplementować odczytywanie i reakcje na stany emocjonalne w środowiskach do tworzenia gier. Z tego właśnie powodu powstała idea opracowania platformy, która w prosty sposób może zostać wcielona do istniejącej już gry stworzonej w określonym oprogramowaniu do tworzenia gier.

3. Specyfikacja komponentów interfejsu do rozpoznawania emocji

Celem pracy jest opracowanie prototypu interfejsu umożliwiającego pomiar sygnałów pozwalających na określenie zmian stanów emocjonalnych gracza. W związku z tym na jeden z elementów niniejszej pracy składa się przegląd możliwych rozwiązań użytych w poszczególnych komponentach interfejsu. W tym rozdziale skupiono się głównie na przedstawieniu urządzeń do pomiaru sygnałów umożliwiających określenie stanu emocjonalnego użytkownika, mechanizmach wnioskowania, które mogą zostać użyte podczas budowania modelu do rozpoznawania emocji, oraz silnikach do budowy gier, przy pomocy których zostanie wykonany końcowy interfejs wraz z grą.

3.1. Platformy pomiarowe

Pomiar sygnałów umożliwiających rozpoznawanie emocji jest jednym z najważniejszych elementów w pętli afektywnej. Obecnie dostępnych jest wiele platform umożliwiających wykonanie takich pomiarów i można je podzielić na dwie główne kategorie: urządzenia służące do odczytu sygnałów fizjologicznych użytkownika oraz sprzęt, z którego można odczytać sygnały pośrednie, na podstawie których można określić stan użytkownika.

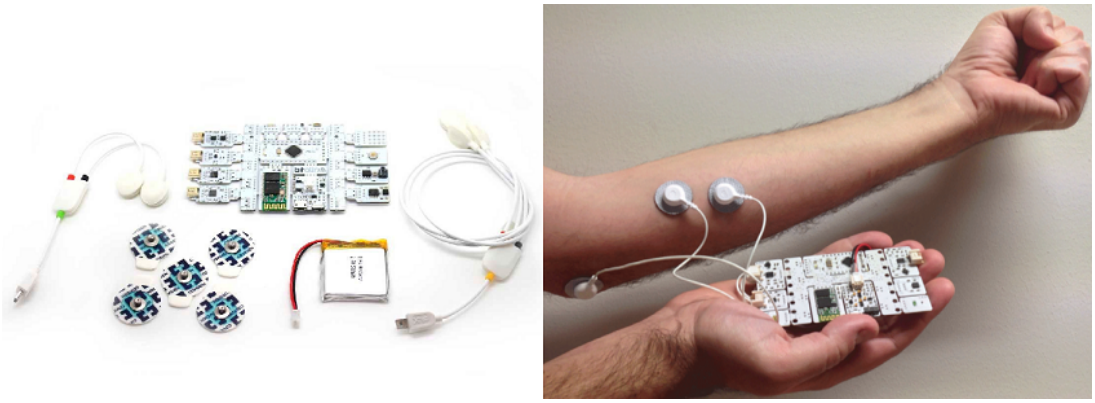
Na pierwszą kategorię składają się między innymi platformy klasy medycznej umożliwiające dokładne pomiary odczytów z ludzkiego ciała. Jednym z takich urządzeń jest Neurobit Optima. Jest to przenośny sprzęt umożliwiający pomiar sygnałów takich jak praca serca, mózgu, ruchów mięśni, reakcji elektrodermalnej czy nawet temperatury skóry [35]. Ogromną zaletą urządzenia są wielofunkcyjne kanały pomiarowe, które użytkownik może dostosować do swoich potrzeb [35]. Atutem, na który warto zwrócić uwagę z perspektywy wykorzystania w informatyce afektywnej, jest gotowe oprogramowanie dostępne od producenta oraz wbudowany interfejs Bluetooth, dzięki któremu urządzenie może być przenośne. Pomiary z Neurobit Optima charakteryzuje wysoka dokładność i stabilność pomiarów. Podobnym do niego urządzeniem jest NeXus-10, który podobnie jak Neurobbit Optima pozwala na pomiar pracy serca i reakcji elektrodermalnej, posiada wbudowany interfejs Bluetooth oraz jest do niego dołączane gotowe oprogramowanie. Niestety ogromną wadą jest waga i rozmiar urządzenia, które wpływają na wygodę podczas użytkowania. W kontekście gier afektywnych potencjalnym słabym punktem urządzeń

Rys. 3.1. Urządzenie Neurobit Optima, źródło: [35]

klasy medycznej może być także ich inwazyjność. Choć w ciągu ostatnich lat środowisko sprzętowe wyszło daleko poza standardowy komputer czy konsole, to część z graczy może nie zaakceptować urządzeń kojarzonych z medycyną jako części stanowiska do gier. Jednym z problemów są tutaj szeroko wykorzystywane elektrody, które, choć umożliwiają dokładne pomiary sygnałów fizjologicznych, są kojarzone jednak głównie ze środowiskiem medycznym.

W kontekście platform do pomiarów sygnałów fizjologicznych w informatyce afektywnej można zauważyć wzrost zainteresowania urządzeniami nasobnymi [36]. Ich wyraźną zaletą jest rozmiar i przenośność, dzięki czemu ruchy użytkownika nie są w żadnym stopniu ograniczone. Najbardziej rozpowszechnionym, a jednocześnie jednym z tańszych rozwiązań, są inteligentne opaski, takie jak Xiaomi Mi Band czy Microsoft Band [36]. Oba z wymienionych urządzeń posiadają wbudowany optyczny sensor pulsu [37, 38], drugie natomiast posiada także sensor do pomiaru reakcji elektrodermalnej [38]. Inną, mniej popularną opaską, jest Empatica E4, która poza sensorami dostępnymi w Xiaomi Mi Band i Microsoft Band posiada sensor do pomiaru temperatury skóry [39]. Warto wspomnieć też o wbudowanych w urządzenia sensorach ruchu używanych między innymi do zliczania kroków mogących posłużyć jako sygnał pośredni, na podstawie którego można wykryć poziom aktywności użytkownika. Niestety ogromną wadą inteligentnych opasek jest ogromna niedokładność pomiarów w porównaniu do odczytów ze sprzętów klasy medycznej. Odczyty z opasek charakteryzują się odczytami mocno odbiegającymi od pomiarów wykonanych przy pomocy sprzętu medycznego [36, 40].

Innymi urządzeniami nasobnymi charakteryzującymi się większą dokładnością, do których często porównuje się odczyty z innych urządzeń do pomiaru pracy serca [36, 40], są monitory tętna Garmin HRM-Run oraz Polar H10. W przeciwieństwie do optycznego sensora wbudowanego w inteligentne opaski są one wyposażone w suche elektrody wbudowane w pasek zakładany na klatkę piersiową[41, 42]. Jest to rozwiązanie, które nie jest inwazyjne dla użytkownika, a jednocześnie pozwala na dokładny pomiar pracy serca. Ważną zaletą obu urządzeń jest nie tylko wsparcie dla standardu Bluetooth, ale także



Rys. 3.2. BITalino (r)evolution kit oraz przykład podłączenia sensora do pomiaru ruchu mięśni, źródło: [35]

dla protokołu ANT+ wykorzystywanego w coraz większej liczbie urządzeń do pomiarów aktywności użytkownika[1], do którego twórcy udostępniają również gotowe implementacje interfejsów do odczytywania pomiarów, dzięki którym w prosty sposób możliwe jest zbudowanie aplikacji interpretujących wysyłane dane.

Rozwiązaniem będącym pomostem między kosztownymi platformami medycznymi a często niedokładnymi urządzeniami nasobnymi jest platforma BITalino (r)evolution kit [43]. Jest to urządzenie modułowe, składające się z płytki stanowiącej rdzeń urządzenia, do której mogą zostać podpięte oddzielne moduły do pomiarów sygnałów fizjologicznych (rys. 3.2). Na liście dostępnych sensorów znajdują się te odpowiadające za pomiar akcji serca, reakcji elektrodermalnej skóry, czynności ruchowej mięśni oraz aktywności mózgu. Każdy z modułów może zostać podłączony do dowolnego kanału urządzenia, natomiast pomiary odbywają się poprzez podłączenie do drugiej strony modułów kabli, na których końcu znajdują się elektrody. Co więcej, poza modułami służącymi do pomiaru reakcji fizjologicznych użytkownika, na liście dostępnych segmentów znajdują się także akcelerometr, sensor światła, dioda LED, brzęczyk, oraz przycisk, które mogą posłużyć do odczytu pomiarów pośrednich czy informowania użytkownika o zdarzeniach. Dzięki temu platforma BITalino może służyć nie tylko jako urządzenie wykorzystywane do pomiarów, ale również do interakcji z grą. Bardzo dużą zaletą BITalino z perspektywy informatyki afektywnej jest dostępność narzędzi przygotowanych przez twórców, a także ogromna ilość implementacji interfejsów do komunikacji z urządzeniem. Twórcy na swojej stronie udostępniają biblioteki dla wielu popularnych języków programowania takich jak Python, C# czy Java, oraz konkretne implementacje z przykładami dla środowisk takich jak silnik do gier Unity. Dodatkowym plusem jest fakt, że większość tych implementacji jest oprogramowaniem otwartym, w związku z czym mogą być one rozszerzane i naprawiane przez społeczność korzystającą z platformy.

Do wspomnianej na początku drugiej kategorii urządzeń, z których możliwe jest odczytanie sygnałów pośrednich wykorzystanych do określenia stanu użytkownika, można przede wszystkim zaliczyć

[1] *https://www.thisisant.com/directory*

sprzęty wykorzystywane przez graczy. Mowa tutaj między innymi o myszkach, klawiaturach czy padach, z których możliwe jest odczytanie intensywności kliknięć lub odczyt szybkości poruszania myszką na podstawie jej pozycji na ekranie [44]. Szczególną uwagę należy poświęcić kontrolerowi DualShock 4 od firmy Sony, który w przeciwieństwie do większości spopularyzowanych kontrolerów do gier posiada wbudowany akcelerometr oraz żyroskop, które mogą posłużyć jako dodatkowe źródło informacji do określenia stanu emocjonalnego użytkownika [45]. Niestety ze względu na umowy licencyjne oprogramowanie umożliwiające odczyt tych parametrów z pada, jest płatne, co można zaliczyć jako wadę tego rozwiązania z perspektywy twórcy aplikacji wykorzystującej funkcjonalności DualShocka.

3.2. Możliwe mechanizmy do rozpoznawania emocji

Aby określić emocje występujące u użytkownika na podstawie danych zebranych przy pomocy wybranych urządzeń, należy zaimplementować mechanizm sztucznej inteligencji, który jak najlepiej będzie umieć określić stan emocjonalny gracza.

Poniżej przedstawiono i opisano kilka wybranych metod wnioskowania, które mogłyby posłużyć jako algorytmy do modelu rozpoznającego stan emocjonalny użytkownika na podstawie zebranych danych fizjologicznych.

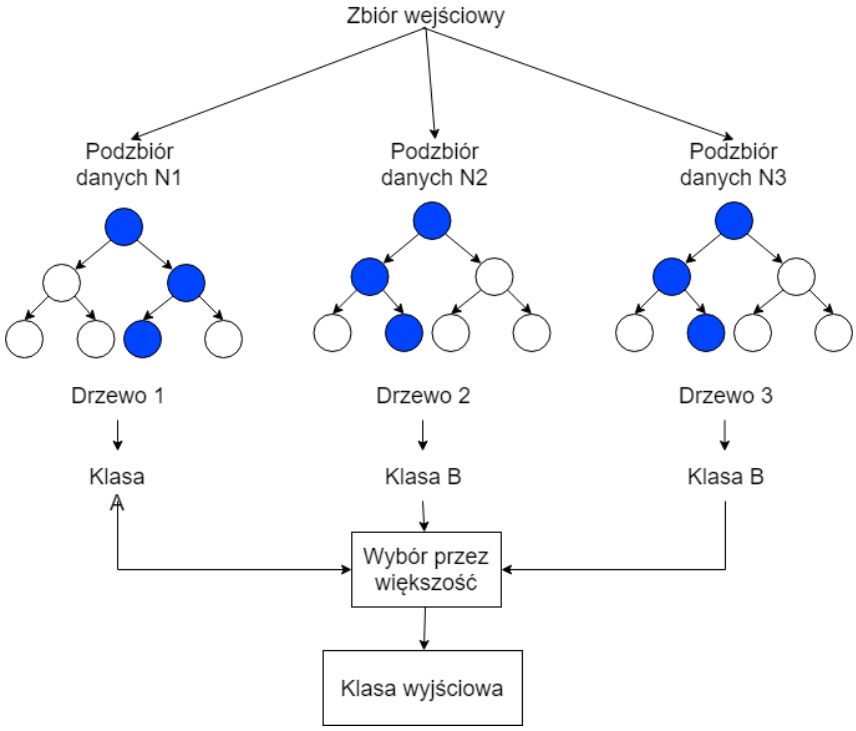
3.2.1. Lasy losowe

Lasy losowe są metodą uczenia maszynowego, która polega na zbudowaniu wielu drzew decyzyjnych, wykonaniu na nich klasyfikacji, a następnie wybraniu klasy poprzez zasadę większości (rys. 3.3). Każde takie drzewo składa się z wierzchołków, z których wewnętrzne nazywane węzłami zawierają pytania i warunki dotyczące sprawdzanych cech, natomiast zewnętrzne wierzchołki nazywane liśćmi zawierają decyzję o klasyfikacji obiektów. Z każdego węzła wychodzi tyle gałęzi, ile jest możliwych wyników warunku sprawdzanego na danym węźle.

Dla każdego z drzew decyzyjnych wybierany jest podzbiór wejściowego zbioru danych za pomocą metody bootstrap, polegającej na losowaniu ze zwracaniem elementów z podanego zbioru [46]. Liczność podzbiorów jest taka sama jak wejściowego zbioru danych, co oznacza, że niektóre elementy mogą znaleźć się w nich więcej niż raz, a inne w ogóle nie zostaną uwzględnione. Dla każdego z drzew wybierany jest także podzbiór cech, które będą brane pod uwagę przy tworzeniu wierzchołków drzewa. W każdym węźle podział jest wybierany na podstawie tego, jak wiele informacji na temat wyboru klasy można uzyskać nam wybrany podzbiór cech.

3.2.2. Drzewa ekstremalnie losowe

Metoda ta jest bardzo podobna do lasów losowych, z dwoma istotnymi różnicami. Pierwszą jest brak tworzenia podzbiorów trenujących dla drzew. Każde z nich jest uczone na tym samym zbiorze danych. Drugą różnicą jest sposób wyboru punktu podziału, który jest zupełnie losowy. Taka metoda



Rys. 3.3. Uproszczony przykład działania metody lasów losowych

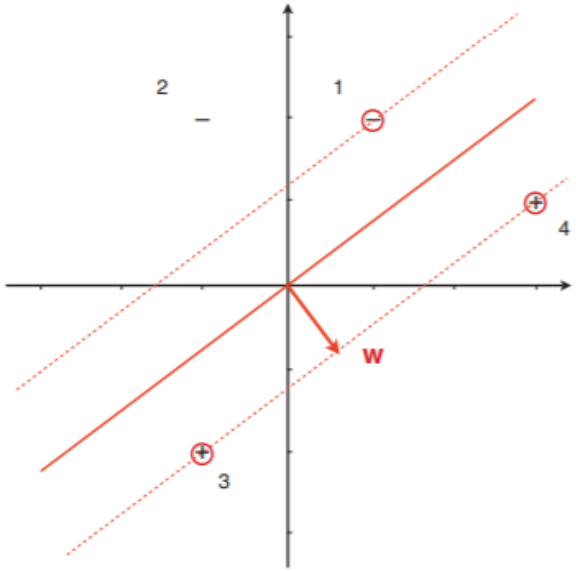
zapewnia dużo mniejszy koszt obliczeń w porównaniu do lasów losowych, ponieważ nie tracimy czasu na sprawdzenie, który z podziałów daje najlepsze rezultaty.

3.2.3. Support Vector Machines

Jest to metoda wykorzystywana głównie w przypadku klasyfikacji. Głównym celem jest znalezienie linii lub hiperpłaszczyzny, która podzieli badany zbiór danych na dwie różne klasy. Ponieważ takich rozwiązań może być nieskończenie wiele, wprowadzone zostało pojęcie marginesu. Poprzez równoległe przesuwanie granicy do pierwszych punktów z obu klas otrzymuje się dwie hiperpłaszczyzny. Marginesem natomiast określa się odległość między nimi. Zadaniem Support Vector Machines jest znalezienie takiej hiperpłaszczyzny, która maksymalizuje margines. (rys. 3.4). Przypadki uczące, na których oparte są równoległe hiperpłaszczyzny, nazywane są wektorami wsparcia.

3.3. Narzędzia do budowy gier

Tworzenie gier komputerowych jest złożonym procesem, wymagającym pracy w ramach wielu dziedzin naukowych. W przypadku przygotowywania wysokobudżetowych produkcji przy pracy nad nimi potrafi pracować nawet kilkaset osób. Od projektantów poziomów, przez osoby zajmujące się przygotowaniem sceny fabularnej, grafików, kompozytorów muzyki aż po twórców oprogramowania, którzy łączą wcześniej przygotowane elementy w jedną całość. To właśnie ci ostatni odpowiadają za ostateczną wersję gry, jej wygląd oraz optymalizację.



Rys. 3.4. Przykład optymalnej linii maksymalizującej margines. Przypadki zaznaczone czerwonymi okręgami są wektorami wsparcia, źródło: [46]

Programiści tworzący końcową wersję gry, podczas tworzenia oprogramowania odpowiadającego za poszczególne jej elementy, muszą wziąć pod uwagę to, z jakim rodzajem gry mają do czynienia. Każdy z gatunków gier charakteryzuje się konkretnymi wymaganiami, które programista musi jak najlepiej odwzorować w tworzonym kodzie gry. Dla przykładu gra wojenna powinna charakteryzować się balistyką pocisków oprogramowaną za pomocą wzorów fizycznych, realistycznym biegiem bohatera, który ma określony poziom wytrzymałości, czy chociażby skutkami wybuchu granatu przedstawionymi jako uruchomione w odpowiednim momencie elementy dźwiękowe i graficzne. W ostatnich latach duzą popularnością cieszą się gry z gatunku Battle Royale [47], w których kilkudziesięciu graczy ściera się ze sobą w rozgrywce wieloosobowej. W przypadku takich gier programiści muszą zwracać uwagę nie tylko na elementy związane z rozgrywką, ale także na jak najdokładniejszą synchronizację klientów gry z serwerem.

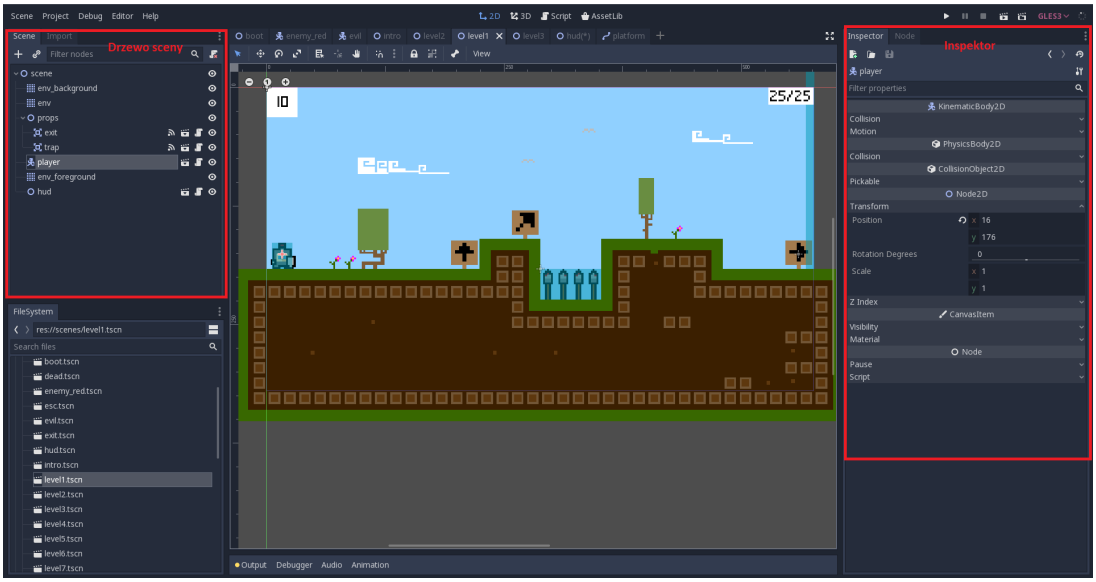
Ilość oraz złożoność elementów, które muszą zostać uwzględnione w kodzie gry, powoduje, że stworzenie jej wyłącznie przy pomocy języków programowania jest niemal niemożliwe. Właśnie w takim celu powstają oprogramowania nazywane silnikami do tworzenia gier. Umożliwiają one obsługę sterowania, grafiki, animacji, interakcji między obiektami, a nawet sztucznej inteligencji, która może być jednym z elementów gry. Silniki zwykle są dostępne w formie środowisk programistycznych, które wprowadzają ułatwienia do zarządzania wcześniej wymienionymi elementami, jednocześnie przy tym nie rezygnując z możliwości języków programowania.

Istnieje mnóstwo silników do tworzenia gier, jednak większość z nich stanowią rozwiązania wykorzystywane jedynie w obrębie danej firmy, która na potrzeby gry lub całej serii stworzyła silnik spełniający wymagania i usprawniający proces tworzenia gier produkowanych przez daną spółkę. Przykładem może być rozwijany od lat IW engine, przy pomocy którego powstała seria gier Call of Duty. Z tego

właśnie powodu większość publikacji naukowych na temat gier komputerowych oparta jest o darmowe silniki do gier. Kilka z takich rozwiązań omówiono poniżej, zwracając uwagę na popularność, funkcje dostępne w danym środowisku, języki, w których można tworzyć aplikacje oraz na jakie platformy można wydać gry stworzone przy pomocy danego silnika.

3.3.1. Godot

Godot [48] jest silnikiem umożliwiającym tworzenie gier na platformy takie jak Windows, macOS czy Linux, a także na środowiska mobilne oraz webowe. Dużą zaletą Godota jest mały rozmiar edytora oraz dostępność na większość popularnych systemów operacyjnych. Skrypty oprogramowujące mechaniki gry mogą być tworzone przy pomocy języków C#, C++, oraz wysokopoziomowego języka GDScript przygotowanego specjalnie dla tej platformy. Alternatywnym rozwiązaniem do tworzenia logiki gry są także skrypty wizualne (ang. *visual scripting*) polegające na budowaniu kodu przy pomocy gotowych bloków, które będą zrozumiałe nie tylko dla programistów, ale również dla grafików czy animatorów. Dużym plusem, szczególnie dla osób, które dopiero zaczynają pracę z Godotem, jest liczba przykładowych projektów, nie tylko w formie gotowych gier, ale również pojedynczych elementów, takich jak sposoby padania światła czy to, w jaki sposób stworzyć realistyczną wodę w grze.



Rys. 3.5. Interfejs silnika Godot, po lewej stronie widać drzewo aktualnej sceny, źródło: opracowanie własne na podstawie [48]

Projektowanie gier przy pomocy silnika Godot opiera się na budowaniu poszczególnych obiektów nazywanych scenami, które mogą następnie być łączone ze sobą na kolejnych scenach. Dzięki temu kolejne poziomy, a nawet cały projekt mogą ostatecznie być opisane w formie grafu powiązań. Dzięki takiemu rozwiązaniu projekty przygotowane przy pomocy Godota są proste w utrzymaniu podczas wspólnej pracy wielu osób. Każdy programista może w danej chwili zająć się oddzielną sceną, nie powodując konfliktów z elementami przygotowanymi przez inne osoby.

3.3.2. Unity

Unity jest silnikiem umożliwiającym tworzenie gier na o wiele większą liczbę platform niż Godot [49]. Poza opisanymi wcześniej Windowsem, Linuxem, macOS'em oraz platformami mobilnymi i webowymi, przy pomocy Unity można tworzyć gry na konsole współczesnej generacji oraz okulary wirtualnej i rozszerzonej rzeczywistości. Sama platforma jest dostępna na systemy Windows oraz Mac, a od niedawna twórcy starają się dostosować ją także na dystrybucje oparte o jądro Linux.

Tworzenie gier w środowisku Unity polega na tworzeniu obiektów, do których przypisane są konkretne właściwości w zależności od rodzaju stworzonego obiektu. Do każdego z takich elementów mogą również zostać przypisane skrypty odpowiadające za logikę działania obiektu, które tworzone są przy pomocy języka C#. W momencie tworzenia niniejszej pracy twórcy zapowiedzieli także dodanie skryptów wizualnych, co stanowi alternatywę dla osób niezaznajomionych z programowaniem lub też z samym językiem. Przygotowane obiekty są umieszczane w środowisku gry nazywanym sceną. Interfejs Unity pozwala na bardzo proste manewrowanie pomiędzy obiektami znajdującymi się na scenie, zarządzenie ich parametrami oraz powiązaniami między obiektami.

Rys. 3.6. Kerbal Space Program, jedna z najpopularniejszych gier stworzonych przy pomocy silnika Unity, źródło: [50]

Unity jest obecnie jedną z najpopularniejszych platform do tworzenia gier komputerowych. Przy jego pomocy powstało wiele znanych na świecie produkcji takich jak gry karciane Hearthstone i Gwint, a także gra symulacyjna Kerbal Space Program (rys. **??**). Popularność ta przejawia się przede wszystkim w postaci ogromnej liczby poradników dostępnych w sieci, a także aktywnej społeczności, która stale komunikuje się między sobą w celu rozwiązania problemów występujących podczas tworzenia gier komputerowych. Dzięki temu, poza standardową dokumentacją, która także jest mocno rozbudowana, użytkownik ma dostęp do dodatkowych informacji uzyskanych od bardziej doświadczonych programistów. Cechą platformy, która mogła wpłynąć na jej popularność, jest także specjalnie przygotowany

Rys. 3.7. Interfejs edytora Unreal Engine w wirtualnej rzeczywistości, źródło: [51]

sklep zawierający zarówno płatne, jak i darmowe dodatki rozszerzające możliwości silnika lub po prostu ułatwiające pracę przy konkretnym gatunku gry.

3.3.3. Unreal Engine

Unreal Engine jest silnikiem, który przez wiele lat był wykorzystywany jedynie do tworzenia dużych, komercyjnych gier. Do momentu wydania wersji trzeciej, projektowanie gier przy jego pomocy było ograniczone odpłatną licencją [51]. Dopiero w roku 2009 twórcy silnika wydali oprogramowanie Unreal Development Kit, będące darmową wersją trzeciej wersji silnika, z uszczuploną bazą gotowych modeli oraz bez dostępu do kodu źródłowego silnika [52]. Dopiero kilka lat po wydaniu Unreal Engine 4 twórcy zdecydowali się na udostępnienie pełnego silnika za darmo.

Spośród opisywanych rozwiązań, Unreal Engine jest najbardziej rozbudowanym. Podobnie do Unity umożliwia tworzenie gier na większość dostępnych platform, w tym także okularów wirtualnej rzeczywistości. Językiem programowania wykorzystywanym do projektowania gier w tym silniku jest C++, co można traktować jako wadę, ponieważ jest to język trudny i wymagający dużo czasu na opanowanie. Z drugiej strony jednak gry napisane przy pomocy C++ potrafią działać szybciej i płynniej od rozwiązań stworzonych przy pomocy języków wysokopoziomowych. Ogromną zaletą silnika, która wyróżnia go na tle innych rozwiązań, jest dostęp do kodu źródłowego. Dzięki temu, twórcy gier mogą dostosować, a nawet rozszerzyć możliwości platformy.

Dla osób niepracujących z kodem Unreal Engine udostępnia rozbudowany interfejs, pozwalający na stworzenie gry bez konieczności pisania kodu. Podobnie jak w przypadku Unity, projekt gry składa się z powiązanych ze sobą scen, tutaj nazywanych poziomami. Na każdej ze scen umieszczone są obiekty, nazywane aktorami, pomiędzy którymi zaprogramowane są interakcje specyficzne dla danej sceny. Osoba tworząca grę może zaprogramować jej logikę przy pomocy skryptów wizualnych. Tutaj

warto opisać kolejny atut wyróżniający ten silnik na tle innych, którym jest możliwość korzystania z edytora przy pomocy okularów wirtualnej rzeczywistości podczas tworzenia gier na tę platformę (rys. 3.7). Twórca dostaje możliwość dokładnego zaprojektowania świata gry, patrząc na nią od razu z perspektywy gracza.

Unreal Engine jest obecnie drugą, tuż po Unity, najpopularniejszą otwartą platformą do gier. Przewaga Unity wynika tutaj nie z większych możliwości silnika, ale z jego otwartości od początku istnienia. Ponieważ jednak przez wiele lat Unreal Engine był wykorzystywany głównie w rozwiązaniach komercyjnych, posiada rozbudowaną, złożoną dokumentację, która opisuje każdy fragment możliwości edytora. Po udostępnieniu darmowej wersji oprogramowania pojawiła się także duża ilość materiałów na temat tworzenia gier przy pomocy Unreal Engine, a także urosła społeczność twórców gier, którzy wykorzystują ten silnik i wymieniają się między sobą cennymi informacjami.

4. Architektura

Na podstawie analizy możliwych rozwiązań sprzętowych w rozdziale 3 oraz przedstawionych poniżej założeń architektury platformy dokonano wyboru urządzeń, które będą stanowiły część sprzętową tworzonego interfejsu. W tym rozdziale zostanie przedstawiona ich specyfikacja oraz argumenty, które przemawiają za wyborem każdego z urządzeń. Opisany zostanie także sposób montażu sprzętu.

4.1. Założenia architektury sprzętowej

Aby wybrać odpowiedni dla przygotowywanego prototypu interfejsu zestaw urządzeń, określone zostały założenia i wymagania, jakie urządzenia powinny spełniać:

1. Urządzenie powinno być lekkie, aby używanie go przez gracza nie powodowało dyskomfortu, który może wpłynąć negatywnie na jakość odbieranych danych.

2. Dokładność sensorów dostępnych w urządzeniu powinna być jak największa, aby mieć pewność odczytywanych zachowań użytkownika.

3. Urządzenie powinno być łatwe w obsłudze, zarówno w kwestii jego zamontowania, jak i uruchomienia odczytów. Powinno być także jak najmniej inwazyjne podczas pomiaru, aby zminimalizować dyskomfort użytkownika.

4. Dla wybranego urządzenia powinno być dostępne oprogramowanie umożliwiające odczyt pomiarów w wybranym środowisku do tworzenia gier. Najlepiej, gdyby rozwiązanie było dostępne w postaci biblioteki oferowanej przez twórcę sprzętu, lub rozszerzenia silnika, które umożliwi odczyt danych z urządzenia.

5. Urządzenie powinno mieć możliwość połączenia bezprzewodowego, najlepiej przy pomocy technologii Bluetooth, lub innego protokołu umożliwiającego bezprzewodowy odczyt pomiarów z urządzenia.

6. Przynajmniej jedno z urządzeń powinno umożliwiać odczyt pracy serca, w postaci pomiaru pulsu lub elektrokardiogramu. Pomiar pracy serca jest jedną z podstawowych metod umożliwiających określenie zmiany w stanie emocjonalnym użytkownika.

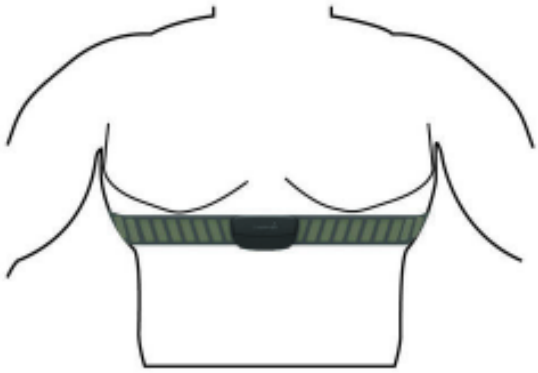
7. Przynajmniej jedno z urządzeń powinno umożliwiać pomiar ruchów mięśni. Odczytane sygnały mogą być wykorzystane do wpływania na rozgrywkę w zależności od aktywności mięśniowej użytkownika w danej partii ciała.

4.2. Garmin HRM-Run

Garmin HRM-Run jest jednym z wielu dostępnych czujników tętna produkowanych przez firmę Garmin. Urządzenie zbudowane jest z modułu zawierającego całą elektronikę odpowiedzialną za interpretację oraz wysyłanie odczytywanych danych. Moduł zasilany jest baterią CR2032 i zamontowany jest na elastycznej opasce, umożliwiającej zamontowanie opaski na klatce piersiowej. Sama opaska zawiera dwie elektrody, które odpowiadają za odczyt sygnałów fizjologicznych, które następnie są przekazywane w formie informacji na temat pracy serca.

Urządzenie, poza podstawowym pomiarem liczby uderzeń na minutę, pozwala na odczyt dodatkowych informacji na temat zmienności rytmu zatokowego w postaci odstępów czasowych pomiędzy kolejnymi tzw. załamkami R, czyli szczytami kompleksów QRS. Na podstawie tej informacji można określić stan zdrowotny użytkownika lub zmiany jego stanu emocjonalnego. Dla przykładu nagłe spadki w zmienności rytmu zatokowego mogą sygnalizować możliwość odczuwania stresu przez użytkownika. HRM-Run umożliwia także odczyt pomiarów dynamiki biegu takich jak liczba kroków na minutę, czas kontaktu z podłożem czy długość kroku. Ze względu na charakter pracy, która skupia się głównie na tematyce gier komputerowych, pomiary te nie były brane pod uwagę.

Urządzenie HRM-Run zostało wybrane, ponieważ stanowi pewien kompromis pomiędzy dokładnością odczytów a rozmiarami i wygodą użytkowania. W porównaniu do inteligentnych opasek, które charakteryzują się najwyższą wygodą użytkowania spośród omówionych w rozdziale 3 urządzeń umożliwiających pomiar tętna, Garmin HRM-Run pozwala na o wiele dokładniejsze odczyty, nie powodując jednocześnie dyskomfortu podczas użytkowania. Jest to możliwe dzięki wykorzystaniu elastycznej opaski ze wbudowanymi suchymi elektrodami, które w przeciwieństwie do fotopletyzmografu odpowiadającego za optyczny odczyt tętna w inteligentnych opaskach, odbierają sygnały elektryczne bezpośrednio z ciała użytkownika, które są następnie interpretowane do danych w postaci wykrycia kolejnych uderzeń serca. Opaski charakteryzują się także dużymi opóźnieniami w stosunku do aktualnego tętna użytkownika, co w przypadku rozpoznawania emocji w sposób ciągły całkowicie eliminuje je jako akceptowalne urządzenia. Jednocześnie, elastyczna opaska, do której przymocowany jest moduł, w bardzo prosty sposób jest zakładana przez użytkownika na klatce piersiowej. Takie umiejscowienie i brak kabli łączących elektrody z głównym modułem sprawia, że użytkownik nie czuje dyskomfortu podczas noszenia urządzenia. W podobny sposób Garmin HRM-Run można porównać z omawianym wcześniej BITalino (r)evolution kit. W tym przypadku można zauważyć sytuację odwrotną, niż przy porównaniu z inteligentnymi opaskami. BITalino oferuje bardzo dokładny odczyt pracy serca, nie tylko w formie liczby uderzeń na minutę, ale pełnego elektrokardiogramu, który umożliwia o wiele szerszą analizę pracy serca. Niestety, odczyt wymaga zamocowania przyklejanych elektrod, które połączone są przewodami z sensorem,



Rys. 4.1. Miejsce montażu czujnika tętna Garmin HRM-Run, źródło: [42]

a następnie z samym BITalino. Taki sposób montażu może wpłynąć negatywnie na komfort podczas gry, co nie wystąpi w przypadku urządzenia HRM-Run.

Innym aspektem, który zadecydował o wyborze tego urządzenia, jest obsługa dwóch protokołów bezprzewodowego przesyłania danych. Poza standardowym połączeniem dzięki technologii Bluetooth HRM-Run wykorzystuje także protokół ANT+ rozwijany przez firmę Garmin. Jego główną zaletą jest brak ograniczeń co do ilości urządzeń, które mogą odczytywać dane z czujnika. W przeciwieństwie do technologii Bluetooth, która ogranicza połączenie do jednego, lub w przypadku Bluetooth Smart do dwóch urządzeń, pomiar przy pomocy protokołu ANT+ umożliwia jednoczesne odczytywanie danych w grze i monitorowanie pracy serca przy pomocy innych urządzeń lub aplikacji korzystających z tego protokołu. Twórcy ANT+ udostępniają szeroką dokumentację oraz biblioteki umożliwiające odczyt w wielu językach programowania, od C++, C#, aż po dedykowane implementacje dla systemu Android.

4.3. BITalino (r)evolution kit

BITalino (r)evolution Plugged Kit BT jest urządzeniem produkowanym przez firmę PLUX Wireless Biosignals przeznaczonym do tworzenia platform, w których zawarte są elementy wymagające informacji na temat sygnałów fizjologicznych. Główny element narzędzia stanowi płytka zbudowana z następujących elementów [43]:

– 10 złącz UC-E6, w ramach których można wyróżnić: 6 wejść analogowych, 1 wejście cyfrowe, 1 wyjście cyfrowe, 1 złącze mogące pracować w trybie wejścia lub wyjścia cyfrowego, oraz 1 złącze PWM, do którego może zostać podłączony przetwornik DAC lub dioda LED. Złącza są podzielone na 2 grupy w oddzielnych segmentach.

– mikrokontroler z mikroprocesorem oraz stykami umożliwiającymi połączenie każdego z segmentów w sposób inny niż standardowy.

– segment zasilający, zawierający włącznik, gniazdo ładownia w formie złącza micro-USB oraz złącze JST, do którego podłączana jest bateria 3.7V zasilająca całą płytkę

– moduł Bluetooth odpowiadający za przesył danych z płytki

W ramach przygotowywanego interfejsu, z dostępnych sensorów opisanych w rozdziale 3 wybrany został jedynie moduł zawierający elektromiogram mierzący napięcie mięśni podskórnych. Głównym powodem takiego wyboru są dwie główne wady urządzenia z perspektywy komfortu użytkowania. Są to przewody łączące płytkę z sensorem, a następnie z elektrodami, oraz same elektrody. Choć z perspektywy eksperymentów naukowych, użycie elektrod żelowych przyklejanych do skóry nie jest problemem, to w przypadku środowiska gier komputerowych, standardowy użytkownik może odczuwać dyskomfort, lub w ogóle zrezygnować z używania urządzenia ze względu na jego inwazyjność.

Głównym powodem wyboru BITalino oraz samego modułu EMG jest wykorzystanie sygnałów fizjologicznych do kontrolowania gry w świadomy sposób. Odczyt zmian napięcia mięśni podskórnych może pozwolić na wykrycie, kiedy i jak mocno są one używane, co może następnie zostać wykorzystane do zaimplementowania mechanik w grze uruchamianych na podstawie konkretnych reakcji mięśniowych. Kolejnym powodem wykorzystania urządzenia jest możliwość sprawdzenia, w jakim stopniu użytkownik tak naprawdę odczuwa dyskomfort podczas używania elektrod przyklejanych do skóry. Pozwoli to stwierdzić, czy urządzenia takie jak BITalino, które umożliwiają dokładny odczyt dzięki wykorzystaniu elementów bliższych rozwiązaniom medycznym, mogłyby przyjąć się jako stały element stanowiska do gier.

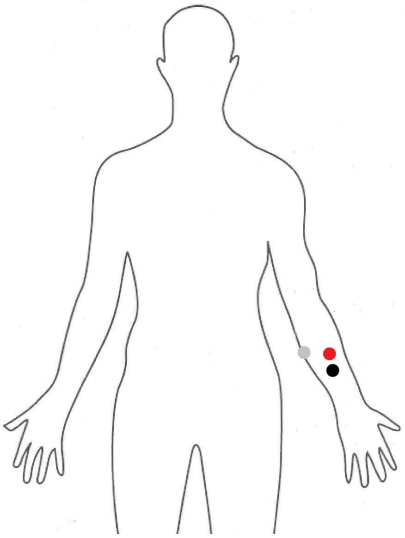
Ponieważ odczyty z elektromiogramu mają posłużyć jako pewien element kontroli nad grą, jako miejsce, do którego podłączony będzie sensor, wybrano przedramię, ponieważ ruch mięśni znajdujących się w tamtej części ciała może być precyzyjny. Dzięki temu użytkownik w prosty sposób, poprzez ruchy nadgarstka lub dłoni, może kontrolować napięcie mięśni w trakcie rozgrywki. Sposób montażu elektrod został przedstawiony na rysunku 4.2. Elektrody czerwona i czarna powinny znajdować się na wewnętrznej części przedramienia, równolegle do niego, stosunkowo blisko siebie. Elektroda referencyjna, w BITalino oznaczona kolorem białym, powinna znajdować się na wystającej kości łokciowej.

4.4. DualShock 4

DualShock 4 jest urządzeniem produkowanym przez firmę Sony, stworzonym początkowo jako kontroler do konsoli PlayStation 4. Na przyciski dostępne na jego powierzchni składają się [45]:

– 2 gałki analogowe, gdzie dla każdej z nich odczyty interpretowane są jako ciągły sygnał wejściowy przedstawiony w dwóch wymiarach.

– Pad kierunkowy (ang. *d-pad*), który stanowi cyfrowy odpowiednik gałki analogowej. Lew i prawy przycisk odpowiadają wartościom granicznym osi horyzontalnej, górny i dolny są przypisane w podobny sposób do osi wertykalnej.



Rys. 4.2. Sposób przypięcia elektrod dla sensora EMG, kolor czerwony oznacza elektrodę dodatnią, czarny ujemną, a szary elektrodę referencyjną

– 9 przycisków cyfrowych, na które składają się: 4 przyciski akcji (trójkąt, kwadrat, krzyżyk, kółko), L3 i R3 zamontowane pod gałkami analogowymi oraz przyciski PS, SHARE i OPTIONS, domyślnie służące do zarządzania funkcjonalnościami konsoli.

– 2 przyciski L1 i R1, dla których wykrycie wciśnięcia opiera się na sile nacisku na przycisk.

– 2 analogowe spusty L2 i R2, których wciśnięcie jest odczytywane jako sygnał w określonym zakresie wartości.

– dwupunktowy panel dotykowy, posiadający także możliwość kliknięcia go.

Innymi elementami widocznymi na kontrolerze są wbudowany głośnik, wejście słuchawkowe Jack 3.5mm, umieszczony na froncie pasek świetlny RGB oraz złącze micro-USB służące do ładowania i podłączenia kontrolera w przypadku braku połączenia Bluetooth. Kontroler, wraz ze wszystkimi opisanymi powyżej elementami, został przedstawiony na rysunku 4.3.

Poza widocznymi elementami kontroler posiada także wbudowane moduły wibrujące, żyroskop oraz akcelerometr. To właśnie dwa ostatnie z wymienionych elementów wpłynęły na wybór urządzenia jako części przygotowywanego interfejsu. W porównaniu do innych dostępnych kontrolerów, poza klasycznym wykorzystaniem narzędzia jako kontrolera do gier akcelerometr wbudowany w DualShocka może posłużyć jako źródło danych, które w pośredni sposób mogą określać pewne elementy stanu emocjonalnego użytkownika. Bardziej pobudzony gracz, może zupełnie nieświadomie poruszać kontrolerem, natomiast brak zmian w odczytach może sugerować, że użytkownik jest skupiony lub znudzony. W porównaniu do określania ilości kliknięć na myszy i kontrolerze w danym czasie, wykorzystanie akcelerometru jest lepszym rozwiązaniem, ponieważ ruchy dłoni gracza trzymającego kontroler są instynktowne w sytuacjach wywołujących pobudzenie. Sceny wywołujące nagły strach, mogą doprowadzić do ruchu

Rys. 4.3. Moduły sprzętowe wykorzystywane w platformie. Od lewej: kontroler DualShock 4, czujnik tętna Garmin HRM-Run, BITalino z modułem EMG

całego ciała, natomiast sytuacje wywołujące złość lub desperację mogą wywołać subtelne zmiany pozycji rąk na kontrolerze, co wywoła zmianę wartości na akcelerometrze. Interpretacja tych zmian oraz odczytów pracy serca pozwoli na dokładniejsze określenie, jaką emocję odczuwa gracz w danym momencie.

Minusem, który warto zaznaczyć w przypadku DualShocka, jest ograniczenie dostępności oprogramowania pozwalającego odczytywać pomiary z akcelerometru. Z powodu licencji narzuconych przez producenta kontrolerów, które domyślnie są przeznaczone dla konsoli PlayStation 4, większość dodatków i bibliotek wykorzystywanych w silnikach do gier jest dostępnych wyłączenie w wersji płatnej, a oficjalne oprogramowanie od twórców jest udostępniane wyłączenie osobom zatwierdzonym przez Sony jako twórcy gier na konsolę PlayStation 4. Jednak pomimo tego ograniczenia, zdecydowano się na wybór urządzenia jako części przygotowywanego interfejsu.

Na rysunku 4.3 przedstawiony został pełen zestaw urządzeń wykorzystanych w przygotowywanym interfejsie. Pozyskane z nich odczyty zostaną wykorzystane do rozpoznania stanu emocjonalnego gracza, a także do kontrolowania mechanik wykorzystywanych w grze. Dokładny opis, w jaki sposób dane będą odczytywane oraz jak będzie wyglądał model do predykcji emocji użytkownika, zostaną opisane w rozdziałach 5 i 6.

5. Mechanizm predykcji emocji

Jednym z zadań niniejszej pracy jest przygotowanie mechanizmu umożliwiającego odczyt emocji na podstawie danych zebranych przy pomocy urządzeń przedstawionych w rozdziale 4. Zdecydowano się na przygotowanie modelu uczenia maszynowego, który na podstawie cech wyciągniętych z sygnałów fizjologicznych będzie w stanie określić stan emocjonalny użytkownika. Aby uniezależnić przygotowywany model od silnika, w którym będzie przygotowywana gra, zdecydowano się przygotować go w języku Python, przy pomocy biblioteki scikit-learn służącej do tworzenia modeli uczenia maszynowego. Komunikacja modelu z grą będzie odbywała się poprzez zapytanie HTTP, które na wejściu powinno przyjmować dane fizjologiczne odczytane z danego zakresu czasu, na wyjściu zwracając reprezentację emocji określoną w modelu.

W tym rozdziale zostanie opisany proces przygotowywania modelu do rozpoznawania emocji. Zostaną opisane wykorzystane zbiory danych treningowych, sposób ich wstępnego przetworzenia, wybór cech i klas emocji, oraz proces uczenia i wyboru najlepszego modelu. Dokładny sposób komunikacji serwera z grą zostanie opisany w rozdziale 6.

5.1. Zbiory danych

W trakcie poszukiwania zbiorów uczących nie znaleziono niestety takich, które jednocześnie wykorzystywałyby pomiary pracy serca oraz akcelerometru. Zdecydowano więc o wyborze zbiorów, które zawierały pomiary pracy serca i to na ich podstawie określony zostanie stan emocjonalny zwracany przez model. Odczyty z akcelerometru zostaną wykorzystane jako dodatkowy kontekst korygujący określony stan emocjonalny, o czym więcej jest wspomniane w rozdziale 6.

5.1.1. DEAP

DEAP[1] [22] to zbiór danych zawierający pomiary przeprowadzone na 32 osobach w wieku 19-37. Każdy z badanych został poddany eksperymentowi, w ramach którego musiał obejrzeć 40 fragmentów teledysków, każdy z nich trwający 60 sekund. Każdy z nich był dostosowany w ten sposób, aby wywoływać konkretny rodzaj emocji. Uczestnicy oceniali każdy z fragmentów w czterech skalach: *valence*

[1] A Database for Emotion Analysis using Physiological Signals

określający poziom przyjemności odczuwanej podczas oglądania, *arousal* opisujący pobudzenie użytkownika, *dominance* wskazujący poziom kontroli nad odczuwanymi emocjami oraz *liking* opisujący jak bardzo materiał wideo podobał się osobie badanej. W trakcie badań uczestnicy byli podłączeni do specjalistycznych przyrządów pomiarowych, przy pomocy których zebrano następujące dane fizjologiczne: elektroencefalogram (EEG), puls objętości krwi (BVP), reakcja elektrodermalną (GSR), elektromiogram (EMG), objętość oddechowa, temperatura skóry oraz elektrookulogram. Poza tym, dla 22 uczestników dostępne były również nagrania twarzy.

Dane udostępnione zostały w dwóch wersjach: oryginalnej oraz wstępnie przetworzonej. Wersja oryginalna zawierała pełne odczyty, próbkowane z częstotliwością 512 Hz. W wersji przetworzonej, próbkowanie zostało zmniejszone do 128 Hz i usunięte zostały 3 pierwsze sekundy stanowiące czas na przygotowanie się uczestników do badania. Część odczytów została także przefiltrowana i uśredniona względem pomiarów wszystkich uczestników.

5.1.2. AMIGOS

AMIGOS[2] [23] jest zbiorem zawierającym dane zebrane podczas dwóch rodzajów eksperymentów:

1. 16 krótkich fragmentów filmów hollywoodzkich wybranych ze zbioru DECAF [53] o długości 51–150 s ocenianych przez grupę 40 osób.

2. 4 długie materiały wideo o długości 14.1–23.58 min ocenianych przez 37 osób.

Podobnie jak w przypadku zbioru DEAP, uczestnicy mieli ocenić fragmenty wideo w kilku skalach: *valence*, *arousal*, *dominance*, *liking* oraz *familiarity* określający poziom znajomości fragmentu wideo. Każdy z badanych miał także zaznaczyć jaką emocję odczuwał podczas oglądania danego materiału, mając do wyboru: emocję neutralną, odrazę, radość, zaskoczenie, złość, strach i smutek. W trakcie badań zostały zebrane następujące dane pomiarowe: elektroencefalogram (EEG), elektrokardiogram (ECG), reakcja elektrodermalna (GSR), nagrania wideo twarzy osób badanych oraz nagrania całego ciała wykonane przy pomocy urządzenia Kinect V1. W przypadku danych z elektrokardiogramu pomiary zostały przeprowadzone na lewym i prawym ramieniu.

Zbiór został udostępniony w dwóch wersjach: oryginalnej, dla której w zależności od odczytywanego sygnału dane są próbkowane z częstotliwością 128 lub 256 Hz, oraz wstępnie przetworzonej. W tej drugiej, próbkowanie wszystkich odczytów zostało zmniejszone do 128 Hz, dane zostały uśrednione względem wszystkich pomiarów, oraz wstępnie przefiltrowane przy pomocy filtrów dolnoprzepustowych.

5.1.3. ASCERTAIN

ASCERTAIN[3] [54] to zbiór danych zebranych podczas badań, w ramach których 56 osób obejrzało 36 materiałów wideo o średniej długości 80 s. Każdy z uczestników oceniał materiały wideo w pięciu

[2] A Dataset for Affect, Personality and Mood Research on Individuals and Groups

[3] a multimodal databa**AS**e for impli**C**it p**ER**sonali**T**y and **A**ffect recognit**I**o**N** using commercial physiological sensors

skalach: *valence* w zakresie od -3 do 3, *arousal* od 0 do 6, *liking*, *familiarity* oraz *engagement* opisujący poziom zaangażowania użytkownika w oglądanie danego fragmentu wideo. Przed przystąpieniem do badania, uczestnicy musieli wypełnić kwestionariusze osobowości BFMS[4]. W trakcie badań zebrano następujące dane pomiarowe: elektrokardiogram (ECG), reakcja elektrodermalna (GSR), elektroencefalogram (EEG) oraz pomiar punktów orientacyjnych twarzy. Podobnie jak w przypadku zbioru ASCERTAIN, pomiar elektrokardiogramu odbywał się na lewym i prawym ramieniu. Częstotliwości próbkowań były zależne od odczytywanego sygnału: EEG – 32 Hz, ECG – 256 Hz, GSR – 128 Hz.

W ramach zbioru danych udostępnione zostały także cechy wyliczone na podstawie zebranych pomiarów, które mogły zostać wykorzystane jako gotowe dane do uczenia modelu maszynowego. Elementem zawartym w zbiorze, który wyróżnia go na tle innych opisanych, jest plik zawierający wyniki wstępnej analizy zebranych danych, które zostały określone w skali od 1 oznaczającego dane perfekcyjne do 6 opisującego brak danych lub błędy w pomiarach. Udostępniona została także lista eksperymentów zakończonych niepowodzeniem. Dzięki temu, w zależności od wymaganej jakości odczytanych danych, można przeprowadzić wstępną filtrację pomiarów.

5.1.4. DECAF

DECAF[5] [53] jest zbiorem zawierających pomiary zebrane podczas eksperymentów przeprowadzonych na 16 mężczyznach i 14 kobietach o średniej wieku 27 lat. Uczestnicy w trakcie badań oglądali materiały z dwóch zbiorów:

1. 40 fragmentów teledysków wykorzystywanych w opisywanym wcześniej zbiorze DEAP [22].

2. 36 fragmentów wyciętych z hollywoodzkich filmów o średnim czasie trwania 80 sekund.

Podobnie jak w przypadku pozostałych zbiorów, uczestnicy mieli ocenić każdy fragment przy pomocy następujących skal: *valence* w zakresie od -2 do 2, *arousal* od 0 do 4 oraz *dominance* o zakresie takim jak w przypadku *arousal*. W trakcie badań zebrano dane o następujących sygnałach fizjologicznych: magnetoencefalogram (MEG), elektromiogram (EMG), elektrokardiogram (ECG). Zebrano także nagrania twarzy każdego z uczestników w bliskiej podczerwieni. Dane były gromadzone z próbkowaniem o częstotliwości 1000 Hz, co daje dużą ilość danych, jednak z drugiej strony może wprowadzać wyraźne błędy w pomiarach. Poza oryginalnymi odczytami, w ramach zbioru zostały udostępnione wyliczone cechy dla każdego z pomiarów, które mogą być wykorzystane jako gotowy, przetworzony zestaw danych wykorzystywanych do stworzenia modelu. Ze względu na generyczny sposób przetworzenia danych na wszystkich używanych zbiorach, zdecydowano się skorzystać jednak z oryginalnych pomiarów.

[4] big-five marker scale

[5] MEG-Based Multimodal Database for Decoding Affective Physiological Responses

5.2. Przetworzenie danych

Ponieważ wykorzystane zbiory danych zostały udostępnione w różnych formatach, a odczyty w nich zawarte różniły się długością pobranych danych, zakresami skal opisujących odczytywane emocje, a w przypadku zbioru DEAP innym rodzajem sygnału opisującego pracę serca, wymagane było przeprowadzenie wstępnego przetworzenia danych wejściowych. Każdy ze zbiorów udostępniał dane w formacie plików MAT, które zostały otwarte przy pomocy biblioteki SciPy[6]. Następnie, dla każdego ze zbiorów przeprowadzono proces przetworzenia danych przedstawiony na rysunku 5.1. Ze względu na dostępność plików zawierającego ocenę jakości danych oraz listę nieudanych eksperymentów, w przypadku zbioru ASCERTAIN został przyjęty dodatkowy krok, który odfiltrowywał przypadki zakończone niepowodzeniem, oraz te, dla których jakość sygnału z elektrokardiogramu była oceniona na 6.

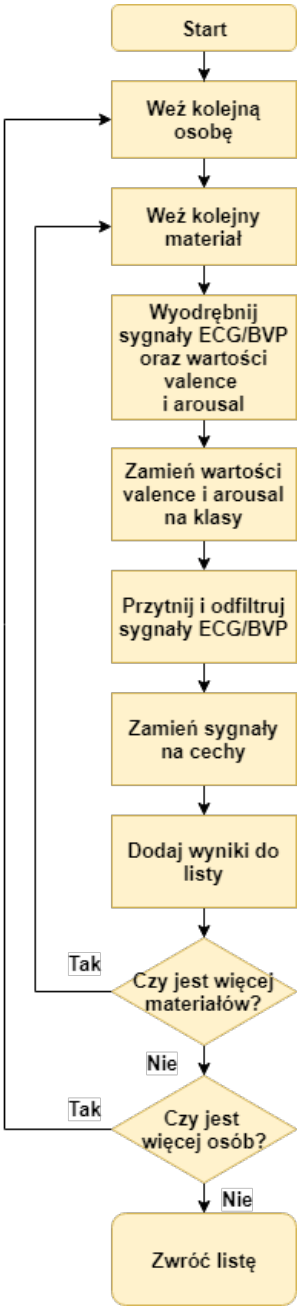
Mając identyfikator osoby i eksperymentu, z wczytanych danych zostały wyodrębnione sygnały z elektrokardiogramu a w przypadku zbioru DEAP z odczytów pulsu objętości krwi. Zostały także odczytane wartości *valence* i *arousal*, na podstawie których określony zostanie cel przygotowywanego modelu. Aby ujednolicić otrzymane wartości ocen, przeskalowano je do zakresu od 1 do 9, a następnie zamieniono na 9 klas, odpowiadających połączeniu 3 poziomów wartości dla każdej z ocen, tak jak pokazano to na rysunku 5.2. Taki podział pozwoli na objęcie stanów emocjonalnych charakteryzujących się niską lub wysoką wartością każdej z ocen, uwzględniając jednocześnie wartości środkowe opisujące stan neutralny. Kolejnym krokiem było dostosowanie sygnałów do przetworzenia ich na cechy. Ponieważ reakcje fizjologiczne, które mogłyby wskazać na zmianę odczuwanej emocji, następują dopiero po jakimś czasie, wczytane dane zostały obcięte do ostatnich 50 sekund eksperymentu. Następnie, aby pozbyć się ewentualnych szumów i wygładzić przebieg sygnałów, przefiltrowano je przy pomocy pasmowoprzepustowego filtru Butterwortha. Częstotliwości graniczne zostały dobrane na podstawie artykułu Fedotova [55] poświęconego określeniu optymalnych wartości tych parametrów podczas filtrowania sygnału z elektrokardiogramu.

Tak przygotowany sygnał został następnie wykorzystany do obliczenia cech, które będą wykorzystane w przygotowywanym modelu. Na początku z odczytanów sygnałów wyciągnięte zostały interwały pomiędzy kolejnymi uderzeniami serca. Aby określić te wartości, w przypadku sygnału z elektrokardiogramu należało znaleźć położenia kolejnym załamków R, natomiast dla pomiarów pulsu objętości krwi kolejne powtórzenia danej fazy pracy serca. Do wykrycia tych punktów wykorzystana biblioteka PeakUtils[7] oferująca algorytm do wykrywania wierzchołków w podanym sygnale. Wymagany parametr minimalnej odległości pomiędzy kolejnych wierzchołkami był wyliczany na podstawie wzoru:

$$\frac{60 * f}{max_{hr}}$$

gdzie f oznaczało częstotliwość danych zbioru, z którego pochodziła próbka sygnału, a max_{hr} przyjętą najwyższą możliwą wartość tętna. Eksperymentalnie została ona ustawiona na 135 uderzeń serca

[6] *https://scipy.org/*
[7] *https://scikit-learn.org/*

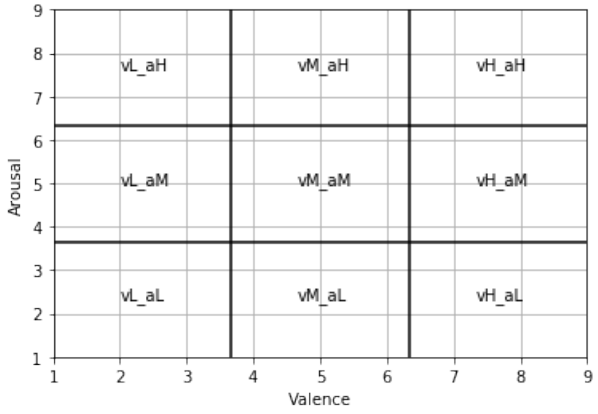


Rys. 5.1. Schemat przedstawiający proces przetworzenia danych do zbioru uczącego

na minutę. Mając na uwadze fakt, że biblioteka mogła wykryć także nieprawidłowe położenia wierzchołków, po wyznaczeniu interwałów przefiltrowano je, odrzucając wartości znacznie odbiegające od mediany z odczytów. Na podstawie odfiltrowanych wartości zostały obliczone ilości uderzeń serca na minutę, a następnie zostały określone następujące cechy statystyczne, które będą brane pod uwagę podczas uczenia modelu:

– Minimalna i maksymalna ilość uderzeń serca na minutę

– Średnia wartość tętna i interwałów między uderzeniami



Rys. 5.2. Podział ocen *valence* i *arousal* na 9 klas

– Odchylenie standardowe tętna

– Kurtoza tętna i interwałów – opisuje, jak bardzo wyniki są skoncentrowane lub oddalone od średniej

– Współczynnik skośności dla tętna i interwałów – opisuje, jak dane rozkładają się wokół średniej.

– Procentowa ilość odczytów tętna i interwałów powyżej wartości: średnia pomiaru + odchylenie standardowe

– Procentowa ilość odczytów tętna i interwałów poniżej wartości: średnia pomiaru − odchylenie standardowe

– Średnie odchylenie bezwzględne (MAD) dla wartości interwałów pomiędzy uderzeniami serca

Analizie została poddana także zmienność rytmu zatokowego, którą rozpatrzono jako parametry opisujące ją w trzech dziedzinach. Pierwszą z nich jest dziedzina czasu, w ramach której można określić następujące parametry:

– odchylenie standardowe interwałów pomiędzy kolejnymi uderzeniami (SDNN)

– średnia kwadratowa różnic pomiędzy kolejnymi interwałami (RMSSD)

– odchylenie standardowe różnic między kolejnymi interwałami (SDSD)

– procentowa ilość interwałów, które różnią się od poprzedniego interwału o więcej niż 20 ms (pNN20)

– procentowa ilość interwałów, które różnią się od poprzedniego interwału o więcej niż 50 ms (pNN50)

Kolejną z dziedzin, w ramach której rozpatrywana jest zmienność rytmu zatokowego, jest częstotliwość. Parametry w ramach tej dziedziny zostały określone poprzez wygenerowanie periodogramu za pomocą

metody Welcha [56] dostępnej jako funkcja w bibliotece SciPy, a następnie wyznaczeniu mocy widma w zakresie określonych częstotliwości [57]:

– 0.04–0.15 Hz – zakres LF, dostarczający informacji o krótkich zmianach w tętnie

– 0.15–0.4 Hz – zakres HF, dostarczający informacji o zmianach związanych z cyklem oddechowym

Jako dodatkowy parametr wykorzystano także stosunek mocy sygnałów LF do HF. Wysoka wartość tej cechy jest widoczna w sytuacjach polegających na walce lub ucieczce, natomiast wartości niskie w przypadku zachowań pozytywnych, takich jak zawarcie przyjaźni.

Trzecią dziedziną, w ramach której można przeprowadzić analizę zmienności rytmu zatokowego, jest dziedzina nieliniowa. W ramach analizy wyznaczany jest wykres Poincaré, który przedstawia korelację pomiędzy kolejnymi interwałami. Oznacza to, że współrzędne każdego z punktów wykresu są opisane jako sąsiadujące wartości interwałów pomiędzy kolejnymi uderzeniami. Cechy wyznaczane w ramach tej analizy to mała i duża oś elipsy okalającej wszystkie punkty wykresu. Mała oś (SD1) jest to odchylenie standardowe odległości punktów od osi $y = x$ i opisuje krótkoczasową zmienność rytmu serca. Duża oś natomiast odpowiada odchyleniu standardowemu odległości każdego z punktów od osi $y = x + avg$, gdzie avg oznacza średnią wartość interwałów. Wartości z tej osi opisują długoczasową zmienność rytmu serca. Jako dodatkową cechę wykorzystano także stosunek SD1 do SD2.

Dla każdego eksperymentu wyciągnięciu wszystkich z wyżej opisanych cech połączono je z wcześniej wyznaczoną klasą i dodano do zbioru uczącego. Przygotowany zbiór następnie przefiltrowano, usuwając te przypadki, w których brakowało przynajmniej jednej cechy. Ze względu na działania wykonywane podczas uczenia i dopracowywania modelu, cechy nie zostały znormalizowane na etapie przetwarzania danych.

5.3. Wybór modelu

Do uczenia modelu na podstawie przygotowanego zbioru danych uczących zostały wykorzystane następujące klasyfikatory z biblioteki scikit-learn[8]:

– RandomForestClassifier

– ExtraTreesClassifier

– SupportVectorClassifier (SVC)

– KNeighborsClassifier

– GradientBoostingClassifier

W pierwszej iteracji wszystkie modele były uczone dla domyślnych wartości parametrów. Następnie, aby sprawdzić czy skuteczności tak przygotowanych modeli są wiarygodne dla danych spoza zbioru

[8] *https://scikit-learn.org/*

Tabela 5.1. Skuteczności wybranych modeli przy określonych parametrach

Algorytm	**Skuteczność dla domyślnych parametrów**	**Skuteczność po przeprowadzeniu walidacji krzyżowej**	**Skuteczność po optymalizacji parametrów**
RandomForestClassifier	28%	27%	35%
ExtraTreesClassifier	31%	29%	36%
SVC	25%	25%	28%
GradientBoostingClassifier	24%	24%	-
KNeighboursClassifier	21%	24%	-

uczącego, przeprowadzono K-krotną walidację krzyżową. Metoda ta polega na podzieleniu zbioru na K podzbiorów, które następnie są wykorzystane do przeprowadzenia K iteracji uczenia modelu, gdzie jako dane uczące wybieranych jest K-1 podzbiorów, natomiast ostatni jest wykorzystywany jako dane testowe. W wyniku przeprowadzenia walidacji zostaje wygenerowanych K modeli o określonych skuteczności, które następnie można wykorzystać do obliczenia średniej skuteczności modelu. W obu przypadkach cechy ostały poddane standaryzacji. Wartości procentowe skuteczności zarówno dla przypadku domyślnego, jak i walidacji krzyżowej przedstawia tabela 5.1.

Aby osiągnąć większą skuteczność, należało przeprowadzić optymalizację parametrów. Zdecydowano się ją wykonać jedynie dla tych modeli, które wykazywały skuteczność wyższą lub równą 25%. Wykorzystana została do tego biblioteka hyperopt-sklearn[9], która pozwala na odszukanie jak najlepszych parametrów modelu w zależności od zadanych danych wejściowych. Funkcja estymująca na wejściu przyjmuje model, dla którego chcemy znaleźć parametry, sposób normalizacji cech, liczbę konfiguracji, dla których ma zostać przeprowadzona analiza oraz maksymalny czas, w jakim każda z prób powinna się zakończyć. Dla każdego z modeli liczba prób została ustalona na 100, natomiast maksymalny czas dla prób ustawiony został na 5 minut. Parametr odpowiadający za normalizację cech został ustawiony na wartość dowolną, dzięki czemu estymator będzie mógł także znaleźć funkcję normalizującą cechy wraz z jej parametrami, która w połączeniu ze znalezionymi parametrami modelu da najlepsze wyniki. Skuteczności modeli po optymalizacji ich parametrów oraz funkcji do normalizacji cech przedstawione zostały w ostatniej kolumnie tabeli 5.1. Największy przyrost, a jednocześnie najwyższą wartość skuteczności wynoszącą 36% można zauważyć dla klasyfikatora wykorzystujące ekstremalne drzewa ekstremalnie losowe. Dla tego modelu została ponownie przeprowadzona optymalizacja parametrów, tym razem dla dłuższego czasu maksymalnego i liczby prób równej 1200, jednak wynik poprawił się zaledwie o 1%.

Taka skuteczność nie pozwala na pewne rozpoznawanie emocji na podstawie odczytów samej pracy serca. Głównym powodem może być tutaj fakt, że czas, na podstawie którego wykrywane były emocje, był na tyle długi, że w trakcie badania użytkownicy mogli odczuwać więcej niż jedną emocję. Ponieważ nie jest jeszcze obecnie możliwe powiązanie wzorców zmian w reakcjach fizjologicznych z konkretnymi

[9] *http://hyperopt.github.io/hyperopt-sklearn/*

emocjami przy pomocy wzorów, nie było także możliwości wyboru odpowiednich fragmentów sygnałów w zależności od eksperymentu. Możliwe, że dołączenie do analizy sygnałów na temat reakcji elektrodermalnej, czy odczytów z elektroencefalogramu pozwoliłoby na uzyskanie wyższej skuteczności, jednak wykorzystanie sensorów do pomiarów tych sygnałów mogłoby wprowadzić wysoki dyskomfort podczas użytkowania, co kolidowałoby z wymaganiami architektury sprzętowej przygotowywanego interfejsu. Skuteczność rozpoznawania emocji przy pomocy wybranego modelu może zostać poprawiona dzięki odczytom z akcelerometru wbudowanego w kontroler. Ostateczny sposób wyliczania i interpretacja stanów emocjonalnych na podstawie przygotowanego modelu oraz pomiarów z akcelerometru zostanie opisana w rozdziale 6.

6. Implementacja

W ramach niniejszej pracy magisterskiej stworzono grę elektroniczną, w której miał zostać wykorzystany przygotowywany prototyp interfejsu do odczytu zmianów stanów emocjonalnych i zachowań gracza. Na podstawie analizy silników do tworzenia gier przeprowadzonej w rozdziale 3 zdecydowano się na wykorzystanie silnika Unity. Głównym powodem takiej decyzji była przede wszystkim ilość materiałów o tematyce tworzenia gier przy pomocy właśnie silnika Unity, a także dostępność rozwiązań, które pozwalały na prostą komunikację z urządzeniami opisanymi w rozdziale 4.

6.1. Podstawowe założenia

Zanim rozpoczęto implementację gry, zostały określone założenia i wymagania implementacyjne i projektowe, według których następnie została zbudowana gra:

1. Gra powinna zawierać mechaniki afektywne, które modyfikują rozgrywkę w zależności od zachowania lub stanu emocjonalnego gracza.
2. Jednym z elementów implementacyjnych powinien być moduł umożliwiający komunikację z urządzeniami opisanymi w rozdziale 4. Moduł powinien także komunikować się z serwerem zawierającym moduł do predykcji emocji oraz w sposób prosty udostępniać zmiany stanu emocjonalnego użytkownika i jego zachowań.
3. Gra powinna mieć możliwość wyboru jednego z dwóch trybów gry: podstawowego, zawierający standardowe mechaniki gry, oraz wersję afektywną.
4. Rozgrywka powinna być powtarzalna, tak aby w trakcie przeprowadzania badań, każdy z uczestników mógł doświadczyć tych samych elementów gry.

6.2. Implementacja gry

Ponieważ przygotowywana gra miała być wykorzystana do przeprowadzenia eksperymentów w celu ewaluacji opracowanego rozwiązania, zdecydowano, że będzie to gra jednoosobowa. W trakcie rozgrywki gracz wciela się w rolę kapitana statku kosmicznego, który musi walczyć z przeciwnikami, aby przeżyć. Jego zadaniem jest sterowanie statkiem i pokonanie jak największej liczby przeciwników przy

pomocy dostępnego wyposażenia. Rozgrywka podzielona jest na poziomy, w których trakcie gracz musi zdobyć określoną liczbę punktów. Na każdym z poziomów dookoła statku generowanych jest kilka rodzajów wrogich statków, których schemat i szybkość generowania są zróżnicowane w zależności od poziomu. Z każdym kolejnym poziomem pojawiają się nowe rodzaje jednostek, które mogą nie tylko wlatywać w gracza lub do niego strzelać pojedynczymi pociskami, ale również wybuchać w jego pobliżu, posiadać większą wytrzymałość lub strzelać innymi rodzajami pocisków. Każdy z przeciwników ma także przypisaną ilość punktów, która jest dodawana do puli po zniszczeniu go. Aby przejść do kolejnego poziomu, gracz musi zebrać odpowiednią liczbę punktów.

Gracz posiada ograniczoną liczbę punktów życia, która zwiększa się po przejściu każdego z poziomów. Aby urozmaicić rozgrywkę, dookoła gracza mogą pojawić się także wzmocnienia w postaci innych broni oraz elementów leczących. Innym elementem mającym mocno wpłynąć na rozgrywkę jest moc specjalna, która umożliwia graczowi wypuszczenie fali w kształcie okręgu, która natychmiastowo niszczy wrogie statki. Ponieważ umiejętność ta bardzo ułatwia rozgrywkę, gracz jest w stanie użyć jej jedynie co określony czas.

Aby wyrównać poziom gry i nie znudzić gracz powtarzalną rozgrywką, każdy z poziomów posiada także dodatkowy tryb, który jest trudniejszy niż podstawowa rozgrywka. Tryb ten jest uruchamiany na pewien określony czas po spełnieniu specjalnych warunków, które w zależności od wersji gry są inne. W trakcie trwania trybu trudnego schemat generowania przeciwników jest zmieniany na ich trudniejsze wersje, posiadające większą ilość życia i inne umiejętności. Zmienia się także szybkość generowania przeciwników, która zostaje podwojona.

Cała gra została stworzona w stylu dwuwymiarowym. Na rysunku 6.1 przedstawiony został fragment przykładowej rozgrywki. Oznaczone elementy interfejsu użytkownika to kolejno:

1. **Pasek wskazujący poziom życia bohatera**. Warto zwrócić uwagę na brak oznaczenia dokładnej ilości punktów życia. Ukrycie tej informacji zmusza użytkownika do kontrolowania poziomu życia w trakcie gry, a także do unikania wszystkich przeciwników, niezależnie od tego, jakie obrażenia zadają.

2. **Ikony aktualnie wybranej broni i dostępności mocy specjalnej**. Służą one przede wszystkim jako elementy informacyjne. Wskazanie aktualnie wybranej broni pozwala graczowi między innymi na zorientowanie się, czy wzmocnienie w postaci innego typu pocisków jest lepsze od aktualnie posiadanego. Ikona mocy specjalnej natomiast jest nie tylko wskazaniem jej dostępności. W trakcie czasu odnowienia mocy specjalnej ikona ta spełnia funkcję orientacyjnego licznika, który wskazuje użytkownikowi, kiedy ponownie będzie mógł z niej skorzystać.

3. **Ilość punktów zdobytych przez gracza oraz pasek postępu dla danego poziomu**. Podobnie jak w przypadku ilości punktów życia, pasek postępu wskazuje jedynie orientacyjny postęp gracza na danym poziomie, dzięki czemu użytkownik nie jest do końca pewien, kiedy ukończy poziom. Z drugiej strony wyświetlana jest suma punktów zdobytych w trakcie całej rozgrywki, co stanowi pewnego rodzaju motywację dla gracza, aby zdobyć jak najwyższy wynik.

Rys. 6.1. Ekran gry z oznaczonymi elementami interfejsu użytkownika

Tak jak zostało wspomniane na początku rozdziału, do implementacji gry wykorzystano silnik Unity. W momencie tworzenia projektu gry zdecydowano się na użycie wersji 2018.3.0f2, która jest przeznaczona do tworzenia gier na systemy operacyjne oparte o architekturę 64-bitową. Ponieważ projektowanie gier przy pomocy tego silnika jest oparte na tworzeniu obiektów, które są wprowadzane w interakcje pomiędzy sobą, a logika za to odpowiadająca zawarta jest w skryptach dołączanych do obiektów, zdecydowano się na stworzenie hierarchii, w której pierwszym poziomem są poszczególne elementy rozgrywki w postaci obiektów tworzonych na scenie gry, drugim natomiast są skrypty odpowiadające za logikę danego obiektu lub innych elementów gry takich jak interfejs użytkownika czy postępy gracza. Do obiektów stanowiących główne elementy świata gry należą:

1. **Player** reprezentujący statek, którym kieruje gracz. Poza elementami graficznymi zawiera on następujące skrypty związane ze sterowanym statkiem:
   - **PlayerController**, opisujący sposób poruszania się kontrolowanego statku. Przy jego pomocy ustawiana są także szybkość poruszania się oraz czas potrzebny na wyhamowanie statku.
   - **PlayerShooter**, odpowiadający za logikę strzelania przy pomocy wybranej broni, a także za użycie mocy specjalnej. Jej głównym zadaniem jest zarządzanie tworzeniem pocisków i aktywacji animacji oraz dźwięków związanych ze strzałem lub aktywacją mocy specjalnej. Skrypt odpowiada także za integrację z elementami interfejsu użytkownika odpowiedzialnymi za wyświetlanie aktualnie używanej broni i dostępności mocy specjalnej. Skrypt umożliwia także zmianę aktualnie używanej broni oraz zarządzanie parametrem czasu odnowienia mocy specjalnej.
   - **Player**, odpowiadający za kontrolowanie życia postaci. Mowa tutaj nie tylko o zmniejszaniu życia po przyjęciu obrażeń lub zwiększaniu podczas leczenia bohatera, ale również ustawianiu nowej wartości maksymalnej ilości punktów życia po zakończeniu poziomu. Skrypt

ten zarządza elementem interfejsu użytkownika odpowiedzialnym za wyświetlanie aktualnej ilości punktów życia. Kontroluje także wywołanie animacji i dźwięków uruchamianych po śmierci bohatera, a także oddelegowanie akcji odpowiedzialnych za zatrzymanie gry do obiektów sterujących rozgrywką.

2. **Obiekty reprezentujące przeciwników**. Każdy z nich różnił się przypisanymi elementami graficznymi oraz skryptami, które mogą różnić się w zależności od typu przeciwnika. Do skryptów sterujących logiką przeciwników należą:

   – **Enemy**, odpowiadający przede wszystkim za zarządzanie poziomem życia przeciwnika, zmniejszaniu go po zadaniu obrażeń przez gracza, zmianie elementów graficznych w zależności od ilości punktów życia przeciwnika, oraz za wywołanie animacji i dźwięków mających nastąpić po śmierci przeciwnika. Skrypt posiada także parametr związany z ilością obrażeń zadawanych podczas zderzenia z graczem i zarządza samą akcją kolizji, zabierając graczowi ustaloną ilość punktów życia.

   – **EnemyMovement**, odpowiadający za sposób poruszania się przeciwnika. W zależności od typu wroga, skrypt zarządza czy ma się on zbliżać do gracza, okrążać go, czy może stać w miejscu. Zawiera on parametry dotyczące szybkości przeciwnika, maksymalnej odległości od gracza, oraz dystansu od niego, na jakim część przeciwników ma się zatrzymać.

   – **EnemyShooter**, odpowiadający za kontrolę strzałów przeciwnika, ich szybkość, stworzenie pocisków oraz wywołanie efektów dźwiękowych strzału. Skrypt ten jest przypisywany wyłącznie do przeciwników typu strzelającego.

   – **EnemyBomber**, zarządzający logiką wybuchu przeciwnika. Kontroluje przede wszystkim zadanie graczowi obrażeń i odepchnięcie innych przeciwników, jeśli znajdują się w zadanej przez parametr odległości. Odpowiada także za uruchomienie animacji oraz dźwięku związanych z odliczaniem do wybuchu.

3. **Obiekty reprezentujące wzmocnienia, które gracz może zdobyć w trakcie rozgrywki**. Poza elementami graficznymi, w zależności od rodzaju wzmocnienia, każdy z nich ma przypisany do siebie następujące skrypty:

   – **PowerUp**, odpowiadający za wywołanie efektu wzmacniającego gracza, oraz ciągły obrót obiektu wzmocnienia. Jest to skrypt abstrakcyjny, co oznacza, że nie jest przypisany bezpośrednio do obiektu, a stanowi bazę dla skryptów, które mogą rozszerzać jego zachowanie. W tym przypadku rozszerzeniem jest funkcja, która odpowiada za nałożenie efektu wzmacniającego na gracza.

   – **HealthPowerUp**, będący rozszerzeniem skryptu PowerUp. W ramach wzmocnienia skrypt przywracał pewną ilość punktów życia określoną przez parametr.

– **WeaponPowerUp**, będący rozszerzeniem skryptu PowerUp. Wzmocnienie w tym przypadku polegało na zmianie broni gracza na tę, która była przypisana do wzmocnienia w formie parametru.

Poza wyżej wymienionymi elementami, które stanowią część gry widoczną dla gracza, stworzone zostały elementy będące częścią menu głównego, a także obiekty zarządzające nieposiadające reprezentacji graficznej. Do każdego z nich został przypisany skrypt, który odpowiadał za kontrolowanie innych elementów rozgrywki:

– **MainMenu**, zawierający funkcje, które uruchamiały oraz zamykały grę. W ramach funkcji inicjującej rozgrywkę skrypt uruchamiał animacje wyświetlające ekran powitalny, a następnie rozpoczynający faktyczną rozgrywkę.

– **PowerUpGenerator**, odpowiadający za generowanie obiektów wzmocnień w trakcie gry. Kontrolował on tworzenie wybranego wzmocnienia w losowo wybranej pozycji w określonej odległości od gracza. Ilość wzmocnień znajdujących się w trakcie gry jest ograniczona, aby użytkownik odczuł, że są to elementy wyjątkowe. Wartość ta, wraz z częstotliwością generowania wzmocnień i odległością od gracza, w jakiej obiekty mają być generowane, sterowane są przez parametry skryptu.

– **EnemySpawner**, wykorzystywany do zarządzania logiką odpowiadającą za generowanie przeciwników w świecie gry. Skrypt ten posiadając określoną w parametrze listę szablonów przeciwników, którzy mają być generowani, co pewien czas tworzy każdego z przeciwników w formie fali. Każdy z nich pojawia się w losowo dobranych miejscach. Częstotliwość fal i odległość, w jakiej przeciwnicy są generowani od gracza, określane są poprzez parametry skryptu.

– **GameManager**, zarządzający zatrzymywaniem rozgrywki, oraz sterowaniem elementami po zakończeniu gry. Skrypt ten kontrolował flagi blokujące wszystkie inne obiekty w przypadku zakończenia gry. Odpowiadał on także za wywołanie animacji wyświetlanych po śmierci gracza, zmianę sceny w przypadku wyjścia z gry, czy jej ponowne uruchomienie w przypadku chęci ponownego rozpoczęcia rozgrywki.

– **Progress**, będący miejscem sterującym postępami gracza. Przy jego pomocy gromadzone są punkty zdobyte w trakcie rozgrywki. Po zdobyciu punktów skrypt sprawdza, czy przekroczony został wynik wymagany do ukończenia poziomu. Jeżeli tak to usuwa wszystkie obiekty przeciwników i pociski ze sceny, a następnie w zależności od tego, czy dany poziom był ostatni, uruchamia funkcje odpowiadające za zakończenie gry, lub przejście do kolejnego poziomu. W przypadku końca rozgrywki, skrypt uruchamia animację wyświetlającą gratulacje dla gracza, a następnie oznacza rozgrywkę jako zakończoną przy pomocy flagi ze skryptu GameManager. W przypadku przejścia na kolejny poziom aplikowana jest nagroda w postaci zwiększonych punktów życia, a następnie ładowany jest kolejny poziom. Jednocześnie, niezależnie od tego, czy gracz ukończył

poziom, aktualizowane są elementy interfejsu wyświetlającego ilość punktów gracz oraz sprawdzany jest warunek wymagany do uruchomienia trybu trudnego gry. Jeżeli tak, to lista generowanych przeciwników zostaje zmieniona na trudniejszą wersję, a częstotliwość fal zostaje podwojona. W ramach parametrów skryptu należy wyróżnić warunki uruchomienia trybu trudnego i czas jego trwania, a także listę poziomów w grze. Ten ostatni parametr stanowi tak naprawdę główny rdzeń rozgrywki, ponieważ zawiera w sobie szablon każdego z poziomów, na który składają się: listy przeciwników, którzy mają być generowani, zarówno w wersji klasycznej, jak i trudnej, częstotliwość fal wrogów, wynik wymagany do ukończenia poziomu, liczba punktów życia dodana graczowi po jego ukończeniu, a także flaga oznaczająca czy dany poziom jest ostatnim.

Wszystkie powyższe elementy stanowiły bazę dla podstawowej wersji gry. Parametry każdego z obiektów zostały dobrane tak, aby gracz odczuwał postęp, który ma doprowadzić go ostatecznie do końca gry. Struktura projektu została zachowana w formie dwóch scen, jednej odpowiedzialnej za menu gry, druga będąca sceną, w której odbywała się faktyczna rozgrywka. Tak jak to zaznaczono podczas opisywania skryptów, całość gry rozgrywa się w jednej scenie. Wynika to przede wszystkim z powtarzalności poziomów, które różnią się wyłącznie parametrami.

Ostatnim etapem tworzenia podstawowej wersji gry było dostosowanie sterowania do wykorzystywanego kontrolera Dualshock 4. Ponieważ Unity dla każdej z gier może mieć zdefiniowaną listę nazw odpowiadającym danym przyciskom lub elementom, które mają pewien zakres działania, poza przystosowaniem standardowych nazw do odpowiednich przycisków na kontrolerze, dodane zostały osie, które zawierały w sobie odczyty z prawego analoga kontrolera. Pozwoliło to na wykorzystanie go do kontrolowania kierunku, w który chciał spojrzeć gracz.

6.3. Odczyt danych fizjologicznych i zmian emocji

Równolegle w trakcie tworzenia implementacji podstawowej gry stworzony został interfejs umożliwiający pobieranie danych fizjologicznych i odczytów akcelerometru z urządzeń opisanych w rozdziale 4. Pobrane informacje wykorzystano do przygotowania prostego interfejsu umożliwiającego implementację mechanizmów oddziaływania na strukturę gry w zależności od otrzymanych danych i tym samym domknięcie pętli afektywnej.

Pierwszym krokiem było przygotowanie skryptów umożliwiających odczyt danych bezpośrednio z urządzeń. W przypadku BITalino (r)evolution kit skorzystano z rozwiązania dostępnego na stronie producenta [58]. Aby uniknąć ewentualnych problemów z kompatybilnością, spośród dostępnych interfejsów programistycznych na platformę Unity, wybrano ten dostosowany do wyższych wersji oprogramowania. Bazując na przykładach oferowanych w ramach interfejsu, stworzony został obiekt BITalino, który w pełni będzie odpowiadał za integrację z urządzeniem. Przypisano do niego następujące skrypty:

– **BitalinoSerialPort**, udostępniony jako część wykorzystanego interfejsu. Skrypt umożliwiał konfigurację parametrów związanych z nawiązaniem połączenia z urządzeniem poprzez transmisję

szeregową, zarówno przez Bluetooth, jak i kabel USB. W ramach parametrów możliwe było ustawienie między innymi limitów czasu oczekiwania na odczyt i zapis danych, jednak najistotniejsza była możliwość wprowadzenia nazwy kanału, przy pomocy którego możliwa była komunikacja z urządzeniem.

– **BitalinoManager**, udostępniony jako część wykorzystanego interfejsu. Skrypt ten umożliwia przede wszystkim konfigurację listy wykorzystywanych kanałów i określenia, jakiego rodzaju dane są przypisane do każdego z nich. W przypadku przygotowywanego projektu ustalony został wyłącznie jeden kanał przyjmujący odczyty z elektromiogramu. Skrypt umożliwia także ustalenie częstotliwości wysyłania danych. Ze względu na chęć uzyskania jak najdokładniejszych odczytów, zdecydowano się na ustawienie najwyższej możliwej wartości, czyli tysiąca próbek na sekundę.

– **BitalinoReader**, udostępniony jako część wykorzystanego interfejsu. Skrypt służy do zarządzania przetwarzaniem danych z płytki, tego, w jakiej postaci są zwracane, oraz jak wielki jest bufor, w którym są one przechowywane. Jego rozmiar ustawiono na sto ostatnich elementów. Wybór takiej wartości wynika przede wszystkim ze względu na chęć uzyskania jak najmniejszego opóźnienia od momentu odczytu danych bezpośrednio z urządzenia do momentu wykorzystania ich przetworzonej wersji. Ponieważ sygnał z elektromiografu ma służyć wykryciu zmian w napięciu mięśni, nie była tutaj wymagana wysoka ilość próbek.

– **SensorController**, przygotowany w ramach niniejszej pracy magisterskiej. Głównym zadaniem tego skryptu był odczyt w czasie rzeczywistym bufora danych, wyznaczanie średniej z wartości bezwzględnych odczytów elektromiografu, a następnie sprawdzenie, czy przez określony czas miało miejsce napięcie mięśni przedramienia. Użycie wartości bezwzględnej wynika z oscylacji występujących w sygnale pobranym z elektromiografu. Wykrycie napięcia mięśni oparte zostało na warunku:

$$|EMG|_{avg} \geq mul \cdot |EMG|_{base}$$

gdzie $|EMG|_{avg}$ jest aktualnie odczytaną średnią wartością, mul mnożnikiem ustawianym jako parametr umożliwiający regulację wykrycia napięcia mięśni, a $|EMG|_{base}$ średnim pomiarem bazowym. Ostatnia z wartości jest wyznaczana podczas fazy kalibracji odbywającej się po rozpoczęciu pomiarów. Przez ustaloną parametrem ilość sekund, w których trakcie użytkownik nie powinien napinać mięśni przedramienia, zbierane są odczyty z elektromiografu, a następnie na ich podstawie obliczana jest wartość służąca jako pomiar referencyjny, z którym porównywane są odczyty w trakcie rozgrywki.

Dodanie czasu, przez jaki miał być napięty mięsień oraz mnożnika w warunku miało na celu odrzucenie krótkich, losowych ruchów ręką, jakie mogły wystąpić w trakcie korzystania z kontrolera. Pozwoliło to także dać użytkownikowi świadomość kontroli nad napięciem mięśni, ponieważ musiał wykonać tę czynność przez określony czas.

Aby obiekty, które mają reagować na napięcie mięśni, mogły w prosty sposób być o tym informowane, został wykorzystany mechanizm zdarzeń z języka C#. W ramach skryptu SensorController przygotowany został delegat, będący typem przechowującym referencję do metody i określającym jakie parametry i typ zwrotny powinna zawierać metoda, której sygnaturę określa delegat. Następnie dodane zostało zdarzenie o typie przygotowanego delegata, które służy jako miejsce rejestracji i wywołania metod, które mają obsłużyć dane zdarzenie. W przypadku skryptu SensorController jest ono wywoływane w momencie wykrycia napięcia mięśni przez określony czas. Tak przygotowany mechanizm pozwala na prawie całkowite uniezależnienie elementów związanych z pomiarami fizjologicznymi od skryptów będących częścią implementacji gry.

Kolejnym krokiem było przygotowanie skryptów do odczytu sygnałów z opaski Garmin HRM-Run i akcelerometru wbudowanego w kontroler Dualshock. Zostaną one następnie wykorzystane w skryptach komunikujących się z modelem opisanym rozdziale 5 i zwracających odczyty emocji użytkownika w trakcie rozgrywki.

Podobnie jak w przypadku odczytów z płytki BITalino (r)evolution kit, do pomiaru tętna wykorzystana została biblioteka do obsługi protokołu ANT+ oraz przykład znajdujące się na stronie producenta [59]. Przy ich pomocy przygotowano następujące skrypty umożliwiające odczyt tętna z opaski:

– **AntReader**, przygotowany na podstawie przykładu udostępnionego przez twórców biblioteki. Kod został dostosowany w taki sposób, aby można było uruchomić odczyt przy pomocy funkcji. Do najważniejszych parametrów skryptu należy przede wszystkim numer i typ urządzenia, z jakiego mają być odczytywane dane. Ponieważ odczyty z urządzenia przychodzą w formie ramki z danymi, do skryptu dodany został parser, który przygotowuje obiekt zawierający informację na temat wartości tętna, ilości uderzeń od rozpoczęcia odczytów, czas ostatniego odczytu, oraz wartość interwału pomiędzy ostatnimi dwoma uderzeniami w milisekundach. Tak przygotowany obiekt jest następnie wysyłany jako parametr zdarzenia, które także zostało dodane jako modyfikacja przykładu. Takie podejście wynika przede wszystkim z protokołu, który bazuje na przesyłaniu zdarzeń w momencie wystąpienia jakiejś akcji.

– **HeartRateManager**, zarządzający danymi przesyłanymi przez skrypt AntReader. Jego głównym zadaniem jest przechowywanie obiektów z informacją na temat pracy serca w buforze, którego rozmiar jest zadany przez parametr. Skrypt ten umożliwia także wyizolowanie z bufora wartości tętna i interwałów pomiędzy uderzeniami w formie tablic.

W przypadku odczytów akcelerometru z kontrolera Dualshock 4 producent udostępnia oprogramowanie wbudowane w silnik Unity, jednak jest ono dostępne wyłącznie dla osób projektujących gry na konsolę PlayStation 4. W związku z tym wykorzystany został odpłatny dodatek Rewired, udostępniający obsługę wielu kontrolerów do gier, nie tylko na poziomie obsługi przycisków, ale również wbudowanych funkcjonalności takich jak kontrola wibracji, oświetlenia, czy odczytów ze wbudowanych w kontroler sensorów, takich jak akcelerometr czy żyroskop. Do odczytów z kontrolera i ich interpretacji przygotowane zostały następujące skrypty:

– **AccelerationReader**, odpowiadający za odczyt i wstępną filtrację danych z akcelerometru. Częstotliwość odczytu jest ustalana za pomocą parametru skryptu, a każdy odczyt jest filtrowany przy pomocy filtru Kalmana, aby pozbyć się ewentualnych szumów.

– **AccelerationHandler**, odpowiadający za przechowywanie i analizę pomiarów zebranych przy pomocy skryptu AccelerationReader. Dane przechowywane są w dwóch listach: jednej przeznaczonej do fazy kalibracji oraz drugiej będącej buforem odczytów z określonej parametrem ilości czasu. Do określenia aktywności użytkownika wykorzystano zryw, czyli zmiana wartości przyspieszenia w czasie. W trakcie fazy kalibracji po zebraniu danych z akcelerometru do przygotowanej listy obliczana była średnia wartość zrywu, która posłuży jako pomiar referencyjny. W ramach przygotowanej funkcji skrypt umożliwia pobranie aktualnej aktywności użytkownika w postaci jednego z trzech stanów: bardzo spokojny, spokojny i podekscytowany. Funkcja oblicza wartość zrywu z aktualnie pobranych pomiarów, a następnie określa czy użytkownik jest podekscytowany na podstawie warunku:

$$jerk_{avg} \geq mul \cdot jerk_{base}$$

gdzie $jerk_{avg}$ jest aktualnie odczytaną wartością zrywu, mul mnożnikiem ustawianym jako parametr umożliwiający regulację poziomu granicznego dla ekscytacji, a $jerk_{base}$ jest pomiarem zrywu uzyskanym w trakcie kalibracji. Następnie, jeśli stan nie został określony jako podekscytowany, sprawdzana jest ostatnio odczytana wartość. Jeżeli gracz był spokojny, a ostatni pomiar odbył się w określonym czasie, to skrypt uznawał użytkownika za bardzo spokojnego. W przeciwnym wypadku zwracany był stan spokojny.

Tak przygotowane skrypty do odczytu i wstępnej analizy danych z opaski do pomiaru tętna i akcelerometru zostały następnie wykorzystane do ustalenia stanu emocjonalnego użytkownika. Umieszczono je w obiekcie EmotionManager, który skupia wszystkie skrypty odpowiedzialne za określenie stanu emocjonalnego użytkownika. Aby powiązać odczytane dane z modelem uczenia maszynowego oraz ustalić emocje odczuwane przez gracza, do obiektu EmotionManager dodane zostały następujące skrypty:

– **ClassifierApiManager**, odpowiadający za komunikację z serwerem zawierającym model rozpoznający emocje. Skrypt wykorzystując dane na temat tętna i interwałów pomiędzy kolejnymi uderzeniami, które znajdują się w skrypcie HeartRateManager, wykonuje zapytania do serwera i zapisuje zwracana klasę emocji oraz parametry, dla których dana emocja została zwrócona. Skrypt zawiera także flagę informującą, czy od momentu ostatniego odczytu emocji przez inne skrypty była ona aktualizowana.

– **EmotionManager**, zarządzający pozostałymi skryptami do odczytywania emocji i będący punktem wyjściowym dla przekazywania stanu emocjonalnego gracza do innych obiektów. Głównym zadaniem skryptu jest uruchomienie pobierania danych z opaski i akcelerometru, a następnie określenie emocji użytkownika na podstawie klasy zwróconej przez model rozpoznający emocję oraz stanu pobudzenia zwracanego przez skrypt AccelerationHandler. W momencie pobierania emocji

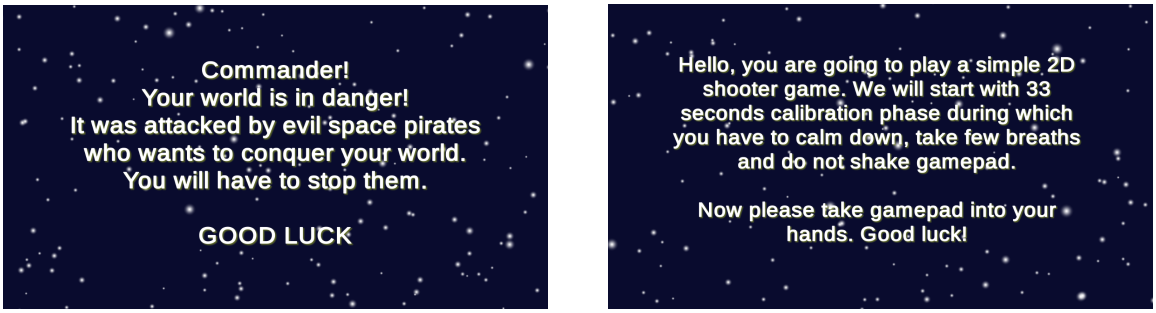
wywoływane jest zapytanie do serwera przy pomocy skryptu ClassifierApiManager o stan emocjonalny użytkownika na podstawie danych z opaski. Zwrócona klasa emocji jest konwertowana do obiektu zawierającego poziomy wartości *valence* i *arousal*, a następnie przy pomocy odczytanego z akcelerometru stanu pobudzenia, są one weryfikowane. Ponieważ pomiar *arousal* jest związany bezpośrednio z pobudzeniem gracza, to właśnie ta wartość była modyfikowana. Jeżeli stan zwracany przez odczyty z akcelerometru wskazywał na pobudzenie użytkownika, poziom *arousal* był zwiększany, natomiast w przypadku bardzo spokojnej gry, był on zmniejszany. Dzięki takiemu podejściu można było łatwo weryfikować klasę zwracaną przez model, która nie zawsze mogła dokładnie odzwierciedlać stan emocjonalny gracza w danym momencie rozgrywki. Po odczytaniu emocji następowało wywołanie zdarzenia o nowej emocji, które jako parametr przyjmowało emocję odczytaną w danym momencie oraz stan poprzedni. Wykorzystanie poprzedniej emocji miało na celu rozszerzenie kontekstu zmian emocji użytkownika i ich interpretację przez konkretne elementy gry. Do najważniejszych parametrów skryptu EmotionManager należą przede wszystkim czas poświęcony na kalibrację oraz parametr określający co ile ma zostać wykonane zapytanie o stan emocjonalny użytkownika.

Obiekty BITalino i EmotionManager zostaly następnie połączone w hierarchii pod obiektem AffectiveManager. Obiekt ten miał służyć jako rdzeń przygotowywanego w ramach pracy magisterskiej interfejsu. Tak przygotowana hierarchia umożliwiała łatwe jego wprowadzenie do istniejącego już kodu gry. Do obiektu AffectiveManager został przypisany także skrypt o tej samej nazwie, którego głównymi zadaniami było uruchamianie oraz wyłączanie mechanizmów opisanych powyżej, a także rozpoczynanie fazy kalibracji dla mechanizmu do odczytywania emocji oraz odczytów z elektromiografu. Dodany został także obiekt Logger, którego skrypt odpowiadał za zapisywanie informacji o odczytanych stanach emocjonalnych oraz momentach wykrycia napięcia mięśni. W przypadku odczytywania emocji zapisywane są poziomy *valence* i *arousal* po weryfikacji odczytami z akcelerometru oraz wartości cech modelu, dla których zostały one wyliczone. Pomiary te mogą posłużyć jako dodatkowe dane uczące, które mogą poprawić skuteczność modelu opisanego w rozdziale 5. W przypadku odczytów z elektromiografu zapisywane były momenty napięcia mięśni przedramienia wraz ze średnią wartością z elektromiografu. Każde zdarzenie było zapisywane w pamięci wraz z czasem jego wystąpienia, a w momencie zakończenia rozgrywki były one zapisywane w pliku CSV.

6.4. Domknięcie pętli afektywnej

Po stworzeniu opisanego w poprzedniej sekcji interfejsu kolejnym krokiem było domknięcie pętli afektywnej. Oznaczało to uzależnienie pewnych elementów gry od stanu emocjonalnego i reakcji użytkownika. Aby to osiągnąć, wymagana była modyfikacja podstawowej wersji gry i zmiana działania jej mechanik w zależności od wartości zwracanych przez skrypty EmotionManager i SensorController.

Aby umożliwić integrację gry z interfejsem, w głównym menu gry przygotowana została opcja, która decyduje o tym, czy mechaniki afektywne mają zostać uruchomione po rozpoczęciu gry. Następnie,



(a) Wprowadzenie w wersji podstawowej (b) Wprowadzenie w wersji afektywnej

Rys. 6.2. Ekran wprowadzenia dla obu wersji gier

w ramach skryptu MainMenu, zmodyfikowany został proces wyświetlania wprowadzenia. W przypadku wersji afektywnej wyświetlony zostaje tekst informujący użytkownika o przeprowadzanej na początku kalibracji interfejsu oraz zasadach, których powinien przestrzegać w trakcie jej trwania (rys. 6.2). Użytkownik proszony jest o uspokojenie się, wzięcie kilku głębokich oddechów oraz nieporuszanie kontrolerem w trakcie przeprowadzania kalibracji. W tym samym momencie uruchamiany zostaje odczyt danych z urządzeń pomiarowych, a po kilku sekundach rozpoczynana jest kalibracja trwająca przez określony w parametrach interfejsu czas. Wcześniejsze uruchomienie odczytu pozwoliło na odrzucenie pierwszych pomiarów z urządzeń, które mogły charakteryzować się dużym błędem ze względu na start urządzeń.

Kolejnym krokiem w integracji interfejsu było przygotowanie mechanik afektywnych, które wpływają na poszczególne elementy rozgrywki w zależności od odczuwanych przez użytkownika emocji lub napięcia mięśni przedramienia. W ramach skryptów odpowiadających za konkretne elementy rozgrywki przygotowane zostały funkcje, które następnie zarejestrowano jako obserwatorzy wystąpienia nowej emocji lub napięcia mięśni. Aby ułatwić interpretację poziomów *valence* i *arousal* zwracanych w ramach obiektu reprezentującego odczuwaną emocję, stworzone zostały stałe, które przypisują każdą z par obu wartości do konkretnej emocji. Interpretacja każdej z kombinacji została przedstawiona w tabeli 6.1. W ramach zaimplementowanych mechanik afektywnych można wyróżnić:

1. **Aktywacja mocy specjalnej po wykryciu napięcia mięśni przedramienia**. Mechanika ta stanowi alternatywę dla uruchomienia mocy specjalnej przy pomocy przycisku na kontrolerze. Uruchomienie jej bez wyłączania akcji z podstawowej wersji gry pozwala użytkownikowi wybrać sposób aktywacji, który jest dla niego wygodniejszy.

2. **Aktywacja mocy specjalnej w momencie, gdy gracz przez dłuższy czas odczuwa złość**. W odróżnieniu od poprzedniego punktu, w tym przypadku moc specjalna aktywowana jest niezależnie od jej dostępności w momencie, gdy poprzednia i aktualnie odczytana emocja to złość.

3. **Zmiana ilości punktów życia gracza na podstawie odczuwanych emocji**. Można wyróżnić tutaj następujące możliwości:

Tabela 6.1. Interpretacja kombinacji poziomów *valence* i *arousal* jako emocje.

Arousal / **Valence**	LOW	MEDIUM	HIGH
LOW	Smutek	Zmęczenie	Zrelaksowanie
MEDIUM	Złość	Emocja neutralna	Szczęście
HIGH	Strach	Zaskoczenie	Ekscytacja

  - Odebranie graczowi 40% punktów życia w przypadku odczuwania przez dłuższy czas emocji neutralnej, zmęczenia lub zrelaksowania. Taka reakcja miała na celu pobudzenie gracza do skupienia się na rozgrywce ze względu na nagłą trudną sytuację.
  - Odebranie graczowi 20% punktów życia w momencie odczuwania przez dłuższy czas szczęścia lub ekscytacji. Emocje te zostały wydzielone do lżejszej wersji negatywnego efektu, aby uniknąć nadmiernego zdenerwowania gracza, ale jednocześnie wzbudzić w nim potrzebę skupienia się na grze.
  - Przywrócenie 10% punktów życia w momencie odczuwania smutku lub strachu. Efekt ten miał posłużyć jako pomoc dla gracza i wywołanie przez to u niego pozytywnych emocji.

4. **Aktywacja dodatkowych fal przeciwników w przypadku odczuwania przez gracza emocji neutralnej, zrelaksowania, zmęczenia lub szczęścia**. W przypadku trzech pierwszych emocji mechanika ta miała na celu postawienie graczowi wyzwania. Uruchomienie jej w momencie odczuwania szczęścia przez gracza miało na celu wywołanie u niego emocji przeciwstawnych takich jak strach czy złość.

5. **Aktywacja i dezaktywacja trybu trudnego gry w zależności od odczuwanej emocji**. Mechanika ta zastępowała jej podstawową wersję opartą na częstotliwości zdobywania punktów. W przypadku odczuwania przez gracza złości lub strachu była ona dezaktywowana, natomiast aktywacja następowała w momencie odczuwania przez gracza emocji neutralnej, zrelaksowania lub zmęczenia.

Implementacja interfejsu do rozpoznawania emocji i zachowań gracza umożliwiła stworzenie gry zawierającej pętlę afektywną. Aby zrealizować w pełni założenia programowania afektywnego, każda z mechanik zmieniała rozgrywkę w taki sposób, aby wzbudzić w użytkowniku inny rodzaj emocji od aktualnie odczuwanych. Przygotowany interfejs umożliwił łatwą implementację każdej z nich bez konieczności modyfikacji oryginalnego kodu gry dzięki rejestracji funkcji uruchamianych w momencie odczytania emocji.

7. Badania

7.1. Procedura eksperymentu

W jaki sposób przebiegał eksperyment

7.2. Uczestnicy

Opis uczestników, wiek, płeć, ewentualny stan zdrowotny

7.3. Analiza wyników

Jakie emocje były odczuwane i jak często, czy uczestnicy byli zainteresowani interfejsem, działaniem samej gry, czy zauważali mechaniki w zależności od odczuwanych emocji

8. Podsumowanie

8.1. Wnioski

8.2. Propozycje przyszłych prac

Bibliografia

[1] Carlos A. Gomez-Uribe i Neil Hunt. „The Netflix Recommender System: Algorithms, Business Value, and Innovation". W: *ACM Trans. Manage. Inf. Syst.* 6.4 (grud. 2015), 13:1–13:19. ISSN: 2158-656X. DOI: *10.1145/2843948*.

[2] O. Amft. „How Wearable Computing Is Shaping Digital Health". W: *IEEE Pervasive Computing* 17.1 (2018), s. 92–98. ISSN: 1536-1268. DOI: *10.1109/MPRV.2018.011591067*.

[3] M. Dikmen i C. Burns. „Trust in autonomous vehicles: The case of Tesla Autopilot and Summon". W: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2017, s. 1093–1098. DOI: *10.1109/SMC.2017.8122757*.

[4] Sebastian Oberdörfer i Marc Erich Latoschik. „Develop your strengths by gaming: towards an inventory of gamificationable skills". W: *INFORMATIK 2013 – Informatik angepasst an Mensch, Organisation und Umwelt*. Red. Matthias Horbach. Bonn: Gesellschaft für Informatik e.V., 2013, s. 2346–2357.

[5] David R. Michael i Sandra L. Chen. „Serious Games: Games That Educate, Train, and Inform". W: (sty. 2006).

[6] Rosalind W. Picard. *Affective Computing*. Cambridge, MA, USA: MIT Press, 1997. ISBN: 0-262-16170-2.

[7] Gartner Inc. *Hype Cycle for Emerging Technologies, 2018*. Spraw. tech. 2018.

[8] I. Kotsia, S. Zafeiriou i S. Fotopoulos. „Affective Gaming: A Comprehensive Survey". W: *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2013, s. 663–670. DOI: *10.1109/CVPRW.2013.100*.

[9] Kristina Höök. „Affective Loop Experiences - What Are They?" W: czer. 2008, s. 1–12. DOI: *10.1007/978-3-540-68504-3_1*.

[10] Jianhua Tao i Tieniu Tan. „Affective Computing: A Review". W: paź. 2005, s. 981–995. DOI: *10.1007/11573548_125*.

[11] Ashley E. Coleman i John Snarey. „James-Lange Theory of Emotion". W: *Encyclopedia of Child Behavior and Development*. Red. Sam Goldstein i Jack A. Naglieri. Boston, MA: Springer US, 2011, s. 844–846. ISBN: 978-0-387-79061-9. DOI: *10.1007/978-0-387-79061-9_3146*.

[12] Walter B. Cannon. „The James-Lange Theory of Emotions: A Critical Examination and an Alternative Theory". W: *The American Journal of Psychology* 39.1/4 (1927), s. 106–124. ISSN: 00029556.

[13] Jesse Prinz. „Emotions Embodied". W: *Thinking About Feeling: Contemporary Philosophers on Emotions*. Red. R. Solomon. Oxford University Press, 2004.

[14] Gerald W. Hohmann. „Some effects of spinal cord lesions on experienced emotional feelings." W: *Psychophysiology* 3 2 (1966), s. 143–56.

[15] SREEJA P S i Mahalakshmi G S. „Emotion Models: A Review". W: *International Journal of Control Theory and Applications* 10 (sty. 2017), s. 651–657.

[16] Keith Oatley i P N. Johnson-laird. „Towards A Cognitive Theory of Emotions". W: *Cognition and Emotion* 1 (mar. 1987), s. 29–50. DOI: *10.1080/02699938708408362*.

[17] Paul Ekman i Wallace V. Friesen. „Constants across cultures in the face and emotion." W: *Journal of personality and social psychology* 17 2 (1971), s. 124–9.

[18] James A. Russell i Lisa Barrett. „Core affect, prototypical emotional episodes, and other things called emotion: Dissecting the elephant". W: *Journal of personality and social psychology* 76 (czer. 1999), s. 805–19. DOI: *10.1037//0022-3514.76.5.805*.

[19] David Watson i Auke Tellegen. „Toward a consensual structure of mood." W: *Psychological Bulletin* 98.2 (1985), s. 219–235. DOI: *10.1037/0033-2909.98.2.219*.

[20] James Russell. „A Circumplex Model of Affect". W: *Journal of Personality and Social Psychology* 39 (grud. 1980), s. 1161–1178. DOI: *10.1037/h0077714*.

[21] James A. Russell, Anna Weiss i G Mendelsohn. „Affect Grid: A Single-Item Scale of Pleasure and Arousal". W: *Journal of Personality and Social Psychology* 57 (wrz. 1989), s. 493–502. DOI: *10.1037/0022-3514.57.3.493*.

[22] Sander Koelstra i in. „DEAP: A Database for Emotion Analysis Using Physiological Signals". W: *IEEE Transactions on Affective Computing* 3 (grud. 2011), s. 18–31. DOI: *10.1109/T-AFFC.2011.15*.

[23] Juan Abdon Miranda-Correa i in. „AMIGOS: A Dataset for Affect, Personality and Mood Research on Individuals and Groups". W: 2017.

[24] Grzegorz Nalepa i in. „Analysis and Use of the Emotional Context with Wearable Devices for Games and Intelligent Assistants". W: *Sensors* 19 (maj 2019), s. 2509. DOI: *10.3390/s19112509*.

[25] Joris H. Janssen, Egon L. van den Broek i Joyce H. D. M. Westerink. „Tune in to your emotions: a robust personalized affective music player". W: *User Modeling and User-Adapted Interaction* 22.3 (2012), s. 255–279. DOI: *10.1007/s11257-011-9107-7*.

[26] Katarzyna Chrzanowska. *Mechanizm rekomendacji muzyki bazujący na wybranych charakterystykach emocjonalnych*. Praca dyplomowa inżynierska. Kraków, 2018.

[27] Charlene Jennett i in. „Measuring and Defining the Experience of the Immersion in Games". W: *International Journal of Human-Computer Studies* 66 (wrz. 2008), s. 641–661. DOI: *10.1016/j.ijhcs.2008.04.004*.

[28] Eva Hudlicka. „Affective computing for game design". W: *4th International North-American Conference on Intelligent Games and Simulation, Game-On 'NA 2008* (sty. 2008), s. 5–12.

[29] Chara Papoutsi i Athanasios Drigas. „Games for Empathy for Social Impact". W: *International Journal of Engineering Pedagogy (iJEP)* 6 (list. 2016), s. 36. DOI: *10.3991/ijep.v6i4.6064*.

[30] Telltale Games. *The Walking Dead: Season Two*. Dostęp: 2018-08-20. 2013.

[31] Grzegorz Nalepa i in. „Affective Design Patterns in Computer Games. Scrollrunner Case Study". W: wrz. 2017, s. 345–352. DOI: *10.15439/2017F192*.

[32] Paweł Jemioło, Barbara Giżycka i Grzegorz Nalepa. „Prototypes of Arcade Games Enabling Affective Interaction". W: maj 2019, s. 553–563. ISBN: 978-3-030-20914-8. DOI: *10.1007/978-3-030-20915-5_49*.

[33] *Horror game Nevermind uses biometrics to sense your fear*. *https://www.gamesradar.com/horror-game-nevermind-uses-biometrics-sense-your-fear*. Dostęp: 2018-08-20.

[34] *Bring to Light*. *https://store.steampowered.com/app/636720/Bring_to_Light*. Dostęp: 2019-08-20.

[35] *Neurobit Optima User Manual*. Neurobit. 2010.

[36] Krzysztof Kutt i in. „Towards the Development of Sensor Platform for Processing Physiological Data from Wearable Sensors". W: czer. 2018, s. 168–178. ISBN: 978-3-319-91261-5. DOI: *10.1007/978-3-319-91262-2_16*.

[37] *Xiaomi Mi Band 3 Manual*. Xiaomi. 2018.

[38] *Microsoft Band Fact Sheet*. Microsoft. 2016.

[39] *Empatica E4 Manual*. Empatica. 2014.

[40] Robert Wang i in. „Accuracy of Wrist-Worn Heart Rate Monitors". W: *JAMA Cardiology* 2 (paź. 2016). DOI: *10.1001/jamacardio.2016.3340*.

[41] *Polar H10 Manual*. Polar. 2019.

[42] *Garmin-HRM Manual*. Garmin. 2015.

[43] *BITalino (r)evolution kit Documentation*. BITalino. 2018.

[44] Jonathan Sykes i Simon Brown. „Affective gaming: measuring emotion through the gamepad." W: sty. 2003, s. 732–733. DOI: *10.1145/765891.765957*.

[45] W: (2013). Dostęp: 2019-09-01.

[46] Peter Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012, s. 215,333. DOI: *10.1017/CBO9780511973000*.

[47] Philipp Moll i in. „A Network Traffic and Player Movement Model to Improve Networking for Competitive Online Games". W: czer. 2018, s. 1–6. DOI: *10.1109/NetGames.2018.8463390*.

[48] *Godot Engine latest documentation*. Godot. 2019.

[49] *Unity User Manual (2019.2)*. Unity. 2019.

[50] *Kerbal Your Enthusiasm. Kerbal Space Program by Squad. https://unity3d.com/showcase/case-stories/squad-kerbal-space-program*. Dostęp: 42019-09-01. Unity, wrz. 2013.

[51] *Unreal Engine 4 Documentation*. Epic Games. 2019.

[52] *Unreal Development Kit Documentation*. Epic Games. 2012.

[53] Mojtaba Khomami Abadi i in. „DECAF: MEG-based Multimodal Database for Decoding Affective Physiological Responses"". W: *IEEE Transactions on Affective Computing* PP (sty. 2015), s. 1. DOI: *10.1109/TAFFC.2015.2392932*.

[54] Ramanathan Subramanian i in. „ASCERTAIN: Emotion and Personality Recognition using Commercial Sensors". W: *IEEE Transactions on Affective Computing* PP (list. 2016), s. 1–1. DOI: *10.1109/TAFFC.2016.2625250*.

[55] Aleksandr Fedotov. „Selection of Parameters of Bandpass Filtering of the ECG Signal for Heart Rhythm Monitoring Systems". W: *Biomedical Engineering* 50 (wrz. 2016). DOI: *10.1007/s10527-016-9600-8*.

[56] P. Welch. „The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms". W: *IEEE Transactions on Audio and Electroacoustics* 15.2 (1967), s. 70–73. DOI: *10.1109/TAU.1967.1161901*.

[57] Fredric Shaffer i Jp Ginsberg. „An Overview of Heart Rate Variability Metrics and Norms". W: *Frontiers in Public Health* 5 (wrz. 2017), s. 258. DOI: *10.3389/fpubh.2017.00258*.

[58] *BITalino (r)evolution kit APIs*. BITalino. 2019.

[59] *Android ANT+ SDK. https://www.thisisant.com/resources/android-ant-sdk/*. Dostęp: 2018-08-20.