AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa magisterska

Opracowanie metod i narzędzi wspierających budowanie gier z użyciem growych wzorców projektowych

Development of methods and tools supporting the game development process using game design patterns

Autor: Paweł Jemioło

Kierunek studiów: Informatyka

Opiekun pracy: dr hab. inż. Grzegorz J. Nalepa

Kraków, 2018

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 profesorowi Grzegorzowi J. Nalepie za poświęcony czas, Łapie za bezinteresowną pomoc i zrozumienie a pozostałym bliskim i rodzinie za zwykłą codzienną cierpliwość.

Streszczenie

Niniejsza praca miała na celu opracowanie zagadnień dotyczących projektowania gier komputerowych na platformie Unity. Zadanie to zrealizowano w oparciu o opis przebiegu tworzenia afektywnej gry elektronicznej. Objaśniono podstawowe pojęcia związane z poruszanym zagadnieniem takie jak interakcja pomiędzy człowiekiem a maszyną, pętla afektywna, growe wzorce projektowe oraz akwizycja i analiza sygnałów fizjologicznych.

Bardzo ważnym elementem pracy było wskazanie sposobu, w jaki udało się wykorzystać dane pochodzące z ludzkiego ciała do oddziaływania na strukturę stworzonej aplikacji. Wykorzystano informację o pracy mięśni dwugłowych ramienia i serca oraz reakcję elektrodermalna, aby oddziaływać między innymi na takie elementy jak szybkość poruszania się wrogów oraz aktywacja specjalnych zdolności bohatera. Rozwiązanie z powodzeniem przetestowano podczas eksperymentów.

Wyrażenie kluczowe: programowanie afektywne, gry afektywne, interakcja między człowiekiem a komputerem, gry komputerowe, emocje, afekt, reakcja elektrodermalna, praca serca, praca mięśni

Abstract

This work was aimed at resolving issues related to the design of computer games on the Unity platform. This task was carried out based on the description of the process of creating an affective game. Basic concepts related to the discussed issue was explained, such as human-computer interaction, affective loop, game design patterns, as well as acquisition and analysis of physiological signals.

A very important element of this work was the indication of the way in which data from human body has been used to influence the structure of the created application. In order to affect elements such as the speed of movement of enemies and activation of the hero's special abilities, among others, information on biceps muscles, heart work and electrodermal reaction was used. The solution was successfully tested during experiments.

Key phrases: affective computing, affective games, human-computer interaction, computer games, emotions, affect, electrodermal activity, heart work, muscle work

Spis treści

Wstęp

Wprowadzenie

W ostatnich latach jesteśmy świadkami fenomenu o wielkim znaczeniu obyczajowym, kulturowym czy wreszcie psychologicznym. Fenomen ten dotyczy zmiany sposobu, w jaki współcześni ludzie postrzegają komputer. Niegdysiejsza „maszyna licząca" nie jest już kojarzona z zakładami pracy, salami uniwersytetów i szkół wyższych. Komputery zarządzają naszymi miastami, przedsiębiorstwami, domostwami, czy wreszcie samochodami. Prawie każda osoba ma telefon komórkowy w zasięgu ręki przez większą część doby. Człowiek cywilizacji Zachodu otacza się tak wielką liczbą urządzeń elektronicznych, że nawet nie zdaje sobie sprawy jak bardzo jest od nich uzależniony [1].

Co istotne, przestaliśmy traktować urządzenia elektroniczne jako narzędzia naszej codziennej pracy, postrzegamy je raczej jako źródło informacji czy rozrywki. Inteligentne asystenty głosowe towarzyszą nam od rana do wieczora i nieustannie nasłuchują [2], gotowe udzielić natychmiastowej odpowiedzi na najbardziej błahe pytania. Urządzenia nasobne monitorują nasze zdrowie [3]. Wszystko po to, aby po nieprzespanej nocy powiadomić nas o tym, że stan naszego snu był słaby i musimy koniecznie coś z tym zrobić [4].

Obraz wyłaniający się z powyższych akapitów tekstu może budzić w czytelniku ambiwalentne uczucia. Z jednej strony można dostrzec automatyzację zadań i powszechny dostęp do informacji przechowywanych w Internecie a z drugiej antyutopijne społeczeństwo wyzbyte wartości wyższych i przywiązane bardziej do swoich telefonów komórkowych [5] i serwisów społecznościowych [6, 7] niż do rodziny i znajomych.

Jak mawiał ojciec medycyny nowożytnej – Paracelsus – „Wszystko jest trucizną i nic nie jest trucizną, bo tylko dawka czyni truciznę" [8]. Te znamienne słowa zdają się mieć znaczenie również w kontekście wykorzystania rozwoju współczesnej informatyki. Szczególnie informatyki jako nauki interdyscyplinarnej, wchodzącej w interakcję z różnymi obszarami wiedzy.

Doskonałym przykładem pola badawczego, skupiającego w sobie teorie naukowe z obszarów, które w klasycznym rozumieniu leżą na przeciwstawnych biegunach poznania, jest informatyka afektywna. Oryginalnie paradygmat tej dziedziny został przedstawiony światu już ponad 20 lat temu [9]. Za główny cel postawiła sobie tworzenie i wykorzystywanie systemów, które zbierają, analizują i przetwarzają dane na temat stanów afektywnych ich użytkowników. Rola tej technologii nie kończy się jednak na biernej stronie interakcji z użytkownikiem, coraz więcej eksperymentów naukowych dotyczy wywoływania emocji w badanych osobach [10]. Co więcej, w 2017 roku informatyka afektywna znalazła się prawie na samym szczycie zestawienia [11] stworzonego przez organizację Gartner w celu przedstawienia technologii, które są obecnie popularne bądź mogą takie się stać w najbliższym czasie wśród badaczy [11].

Waga informatyki afektywnej jest więc szczególna właśnie ze względu na to, że dotyczy zagadnień związanych z ludzkimi emocjami. Należy tu zaznaczyć, że przez lata emocje uchodziły za nierozsądny komponent ludzkiej psychiki toczący odwieczny bój z rozumem działającym w oparciu o racjonalne zasady [12]. To stwierdzenie stoi jednak w sprzeczności z odkryciami dotyczącymi inteligencji emocjonalnej. Wielokrotnie wykazywano, że ludzie cechujący się wysokim wskaźnikiem tej cechy odnoszą sukcesy w różnych dziedzinach życia – od sektora prywatnego [13], do usług publicznych [14]. Ponadto, emocje ułatwiają, czy wręcz umożliwiają podejmowanie słusznych decyzji [15] oraz odgrywają kluczową rolę w kampaniach reklamowych [16] o największej skuteczności medialnej.

Kolejnym przykładem interdyscyplinarnego obszaru wykorzystującego nowe technologie są gry komputerowe. Na proces ich tworzenia składają się między innymi: wymyślenie fabuły gry, osadzenie jej w konkretnym miejscu i czasie, określenie mechanik umożliwiających przyszłemu graczowi interakcję z wykreowanym światem, opracowanie szaty graficznej, nagranie głosów bohaterów, wyprodukowanie muzyki, czy wreszcie stworzenie oprogramowania, które spina wymienione wcześniej elementy w całość. Gry znajdują zastosowanie nie tylko na polu rozrywkowym, ale także w postaci gier poważnych [17], powstających w celu szkolenia strażaków, policjantów, żołnierzy i antyterrorystów oraz edukacji dzieci.

Należy też zwrócić uwagę na próbę połączenia dziedziny informatyki afektywnej z grami komputerowymi. Powstała w ten sposób podgałąź nosi nazwę gier afektywnych. Jej obszarem badawczym jest mierzenie i analiza stanów emocjonalnych inicjowanych poprzez gry lub takie dostosowywanie oprogramowania w czasie rzeczywistym, aby było możliwe zwiększenie doznań towarzyszącym graczom [18]. Istotnym zagadnieniem tej dziedziny informatyki afektywnej jest personalizacja aplikacji z uwzględnieniem cech indywidualnych użytkowników i dywersyfikacja scenariuszy growych w oparciu o upodobania graczy.

W niniejszej pracy położono nacisk na bardzo istotny element współczesnych rozwiązań z zakresu programowania gier ingerujących w emocje, jakim jest wykorzystanie pętli afektywnej [19]. Informacje na temat zachowania i stanu emocjonalnego danej osoby są mierzone przy pomocy założonych czujników, bądź wnioskowane pośrednio ze stanu aplikacji. Po wykryciu interesujących zmian fizjologicznych, za pomocą zdefiniowanych reguł bądź z wykorzystaniem metod sztucznej inteligencji, stan gry jest aktualizowany.

Ponadto w pracy skupiono się na rozwinięciu koncepcji afektywnych growych wzorców projektowych, ułatwiających rozpoznawanie i wywoływanie emocji związanych z konkretnymi mechanikami użytymi w grze [20, 21, 22]. Pomysł ten wywodzi się bezpośrednio od growych wzorców projektowych [23], które znacznie wspomagają złożony przebieg projektowania gier.

Bardzo istotne jest pokazanie, że założenia programowania afektywnego mogą być realizowane z wykorzystaniem sprzętu nielaboratoryjnego. W dotychczasowej literaturze brakuje pozycji, która omawiałaby system implementujący pętlę afektywną w oparciu o urządzenia nasobne i dodatkowo odnoszący się do zagadnień inżynierii oprogramowania poprzez afektywne growe wzorce projektowe. Niniejszy dokument stanowi próbę uzupełnienia luki w tej części informatyki.

Struktura pracy

Praca składa się z siedmiu kolejno ponumerowanych rozdziałów, które są podzielone na odpowiednie mniejsze sekcje. Każda z sekcji opisuje inne zagadnienie praktyczne lub teoretyczne.

W rozdziale pierwszym przedstawiony został aktualny stan wiedzy dotyczący zagadnień, które wiążą się z interakcją pomiędzy człowiekiem a komputerem. Szczególny nacisk położono na dziedziny programowania afektywnego oraz gier afektywnych, przedstawiając wybrane publikacje, problemy oraz kierunki badań z tego zakresu.

Rozdział drugi zawiera informacje związane z procesem projektowania gier komputerowych. Zwrócono uwagę na mocne i słabe strony powszechnie wykorzystywanych narzędzi, które wspierają budowanie tego rodzaju oprogramowania. Ważnym punktem tej sekcji jest odwołanie się do inżynierii oprogramowania poprzez opis wzorców projektowych wykorzystywanych w trakcie tworzenia gier komputerowych. Dodatkowo wyszczególniono również sekcję dotyczącą afektywnych growych wzorców projektowych, gdyż dziedzina ta jest istotna z punktu widzenia tematu niniejszej publikacji.

Następna część pracy opisuje aktualny stan wiedzy na temat sposobów monitorowania aktywności oraz wnioskowania na temat emocji użytkownika. Wyszczególniono metody wykorzystujące pomiary bezpośrednie i pośrednie.

W rozdziale czwartym zawarto charakterystykę zakresu stworzonego projektu. Jest to swego rodzaju przejście od koncepcji teoretycznych do zagadnień natury praktycznej.

Rozdział piąty stanowi jedną z najważniejszych części niniejszej pracy. Przedstawiono w nim opis utworzonego oprogramowania – gry komputerowej z zamkniętym sprzężeniem zwrotnym (pętlą afektywną). Skupiono się na pokazaniu jak za pomocą metod inżynierii oprogramowania takich jak growe wzorce projektowe wykonano projekt aplikacji. Ta sekcja zwiera dokładny raport z przebiegu procesu implementacji afektywnej gry komputerowej w środowisku Unity. Omówiono tam również aspekty związane z monitorowaniem fizjologii gracza. Następnie przedstawiono sposób, w jaki elementy te ze sobą oddziałują by sprzężenie zwrotne mogło być zamknięte.

W rozdziale szóstym zaprezentowano opis środowiska testowego, który umożliwił ewaluację oprogramowania oraz charakterystykę eksperymentów, które zostały wykonane.

Ostatnia sekcja zawiera podsumowanie, wnioski płynące z wykonania zadania projektowego oraz jego mocne i słabe strony. Zwrócono również uwagę na możliwe kierunki, które mogą zostać podjęte w celu dalszego rozwoju projektu.

Na końcu pracy znajduje się bibliografia, zawierająca wszystkie wykorzystane w pracy pozycje z literatury. Za tym spisem umieszczono dodatek A – fragmenty skryptów, które umożliwiają poprawne działanie gry, dodatek B – instrukcję dla osoby badanej, dodatek C – kwestionariusz po ukończeniu eksperymentu oraz dodatek D – wykaz elementów znajdujących się na dołączonej płycie CD.

1. Człowiek a komputer

1.1. Interakcja

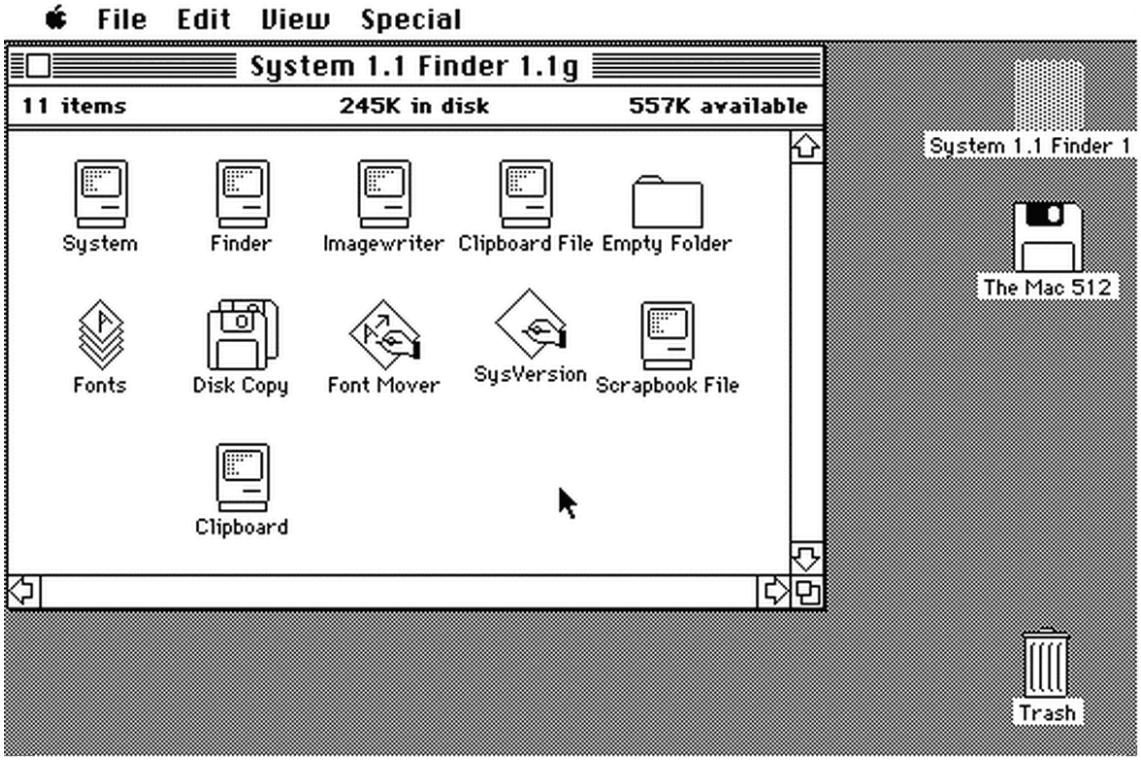
Komunikacja stanowi jeden z najbardziej podstawowych a zarazem najważniejszych procesów towarzyszących człowiekowi od początku istnienia. Niektórzy badacze są wręcz zdania, że jest podstawą sukcesu ewolucyjnego, który udało się osiągnąć *Homo Sapiens* [24]. Nie dziwi zatem, że podstawowe koncepcje dotyczące komunikacji międzyludzkiej zostały przeniesione do interakcji ze sprzętem elektronicznym. Wraz z postępującym rozwojem można zauważyć, że to oddziaływanie zaczyna przybierać coraz bardziej ludzką formę.

Początkowo nie przywiązywano zbyt dużej wagi do roli interakcji między komputerem a człowiekiem. Maszyny liczące były przeznaczone do tego, by być obsługiwanymi przez wyspecjalizowanych inżynierów bądź ludzi skupionych wokół uczelni technicznych [25]. Problem pojawił się wraz z popularyzacją rozwiązań informatycznych [26], gdy ludzkość zaczęła wymagać łatwiejszego sposobu porozumiewania się z komputerami.

Zastąpienie podstawowych struktur elektronicznych sieciami połączeń tranzystorów umożliwiło stworzenie systemów programowalnych opartych o polecenia tekstowe [27]. Skomplikowana aparatura – diody, przetworniki i pokrętła zostały zastąpione układem składającym się z klawiatury ze standardem QWERTY i wyświetlacza [28]. Aby obsługiwać komputer, należało nauczyć się pewnej liczby komend [29] oraz umieć reagować na to, co było wypisywane na ekranie. Z punktu widzenia użytkownika zmiana ta była drastyczna, wciąż jednak nie na tyle, aby umożliwić upowszechnienie się systemów informatycznych w domach przeciętnych ludzi [29]. Co jednak istotne, profesjonalni programiści do dzisiaj korzystają z rozwiązań, które oferuje interfejs tekstowy [30, 31].

Rewolucja w zakresie interakcji z komputerem nadeszła wraz z interfejsem graficznym jeszcze w latach 70. XX wieku dzięki firmie Xerox [32]. Późniejsze działania firmy Apple (rys. 1.1) oraz Microsoft doprowadziły do stworzenia całych systemów operacyjnych [32], które wykorzystywały efekty wizualne. Użycie interfejsu graficznego miało swoje podstawy w psychologii

poznawczej, gdyż największą część danych docierających do człowieka stanowią dane wzrokowe. Co więcej, większość ludzi łatwiej zapamiętuje oraz preferuje pracowanie właśnie z danymi o treści wizualnej [33]. Nie może zatem dziwić, że interfejs graficzny przyczynił się do popularyzacji komputerów, co z kolei wywołało rewolucję urządzeń osobistych.



Rysunek 1.1. *Pierwszy system operacyjny stworzony przez firmę Apple*, źródło: [34]

Interfejs oparty o obrazy spowodował upowszechnienie się nowych urządzeń takich jak myszy komputerowe [32], a rozwój technologiczny z początku XXI pozwolił na masowe wykorzystanie zmysłu dotyku w oddziaływaniu ze sprzętem poprzez specjalne ekrany i touchpady o wysokiej czułości [35, 36]. Jest to o tyle istotne, że to właśnie dotyk jest jedną z ważniejszych i intymnych form komunikacji znanych ludziom.

Rozwój sztucznej inteligencji umożliwia wykorzystanie innych sposobów do oddziaływania na urządzenia elektroniczne i systemy, które są na w nich zaimplementowane. Przechwytywanie, analiza i wreszcie rozpoznawanie mowy zbliżone do ludzkiej [37] pozwala na wydawanie poleceń wirtualnym asystentom [38]. Co więcej, twórcy oprogramowania nie zapominają o tym, że komunikacja to proces dwustronny [39]. Asystenci potrafią zatem odpowiadać na zadane pytania, opowiadać zabawne historie, czy nawet dzwonić w celu umówienia wizyty w salonie fryzjerskim bądź restauracji [40].

Mowa jest o tyle istotna, że wirtualni pomocnicy pojawiają się w coraz większej liczbie domów w coraz większej liczbie urządzeń. Tworzone są urządzenia Internetu rzeczy, które używają komunikacji bezprzewodowej by pomagać ludziom w codziennych czynnościach [41]. Ważne jest zatem, żeby przekazywanie poleceń oraz informacji odbywało się w formie, która jest jak najbardziej naturalna.

Rysunek 1.2. *Wykorzystanie okularów Microsoft HoloLens do edukacji*, źródło: [42]

Innym obszarem, rozwijanym równolegle obok rozpoznawania mowy jest alternatywna rzeczywistość. Według klasycznego podziału może mieć ona formę rozszerzoną, wirtualną bądź mieszaną [43]. Dzięki inteligentnym okularom i hologramom komunikacja z systemami nabiera bardziej immersyjnego charakteru. Możliwe jest natychmiastowe przenoszenie się w miejsca oddalone o tysiące kilometrów (rys. 1.2), obcowanie z dziką naturą, czy walka z bestiami rodem z powieści fantasy [44]. Wszystko to oczywiście z uwzględnieniem wygody domowego zacisza. Jest to kolejny przykład na to, że wraz z postępem technologicznym interakcja z komputerem jest uproszczana do postaci jak najbardziej przypominającej bezpośredniość komunikacji pomiędzy dwiema osobami [45].

Co więcej, ludzie próbują zacieśnić oddziaływanie z komputerami poprzez tworzenie interfejsów bezpośrednio współpracujących z mózgiem [46]. Jest to skomplikowane zadanie wymagające zrozumienia działania umysłu na wielu płaszczyznach. Następnie konieczne jest umożliwienie takiego kodowania i dekodowania sygnałów pochodzących z maszyn, aby mogły być bezpośrednio interpretowane już w mózgu oraz analogicznie w drugą stronę.

Jest naturalną koleją rzeczy, że ten ogromny postęp technologiczny może budzić pewne obawy, szczególnie wśród starszych użytkowników. Obcowanie z robotami o humanoidalnym kształcie i pomysł podłączania urządzeń elektronicznych bezpośrednio do mózgu wciąż odbierane są z dozą rezerwy, czy wręcz jawną niechęcią. Nie pomagają tutaj współczesne autorytety, które w sztucznej inteligencji dopatrują się wroga ludzkości [47, 48].

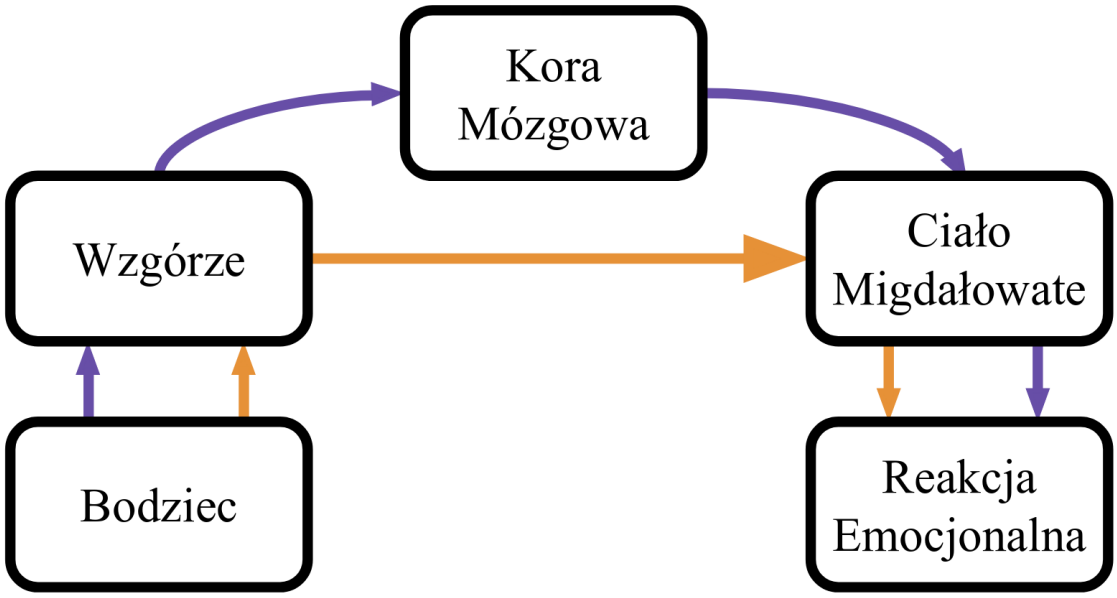
Pewną nadzieją na uczłowieczenie współczesnych systemów elektronicznych i interakcji z nimi daje wykorzystanie emocji do interakcji. Wspomagają one ludzi przez cały czas w komunikacji interpersonalnej [39, 45]. Rzadko kiedy możemy z całą pewnością stwierdzić, że danej osobie nie towarzyszą jakieś uczucia. Z łatwością umiemy odczytywać emocje innych, dostrajać się do nich, czy nawet wykorzystywać je do własnych celów. Ponadto, mimo powszechnego przekonania o irracjonalności [12], uczucia niejednokrotnie pomagają ludziom w codziennym życiu. Na przykład przy podejmowaniu decyzji [15]. Zachęceni tymi psychologicznymi spostrzeżeniami, informatycy coraz śmielej wykorzystują postęp technologiczny, by wnieść interakcję z systemami i maszynami na ten sam poziom.

1.2. Emocje

Emocja jako samodzielny byt psychiczny odgrywała niebagatelną rolę od samego początku istnienia współczesnej psychologii. Już William James [49], znany ze swojej szerokiej wiedzy i błyskotliwych intuicji, zdawał się sugerować, że wraz z uczuciem zawsze pojawia się komponent fizjologiczny. Mimo zgody, że afekt jest nieodłączny emocji, umiejscowienie tego stanu w procesie indukcji uczuć, spędzało sen z powiek późniejszym neuropsychologom [50]. Niektórzy wnioskowali, że element o podłożu organicznym powstaje niezależnie od poznania [51], inni wprost przeciwnie sugerowali, że dopiero uświadomienie sobie jakiegoś zjawiska może skutkować pojawieniem się emocji [52].

Opracowania Zajonca przyniosły swego rodzaju przełom w strukturze badań psychologicznych. Naukowiec zasugerował, że występują dwie niezależne drogi indukcji emocjonalnej [53]. Pierwsza, związana bezpośrednio z oceną poznawczą, i druga, w której pierwszeństwo ma zmiana w ciele. Jego badania zasługują na uwagę ze względu na to, że kilka lat później LeDoux dostarczył ich potwierdzenia, odnajdując w mózgu dwie drogi (rys. 1.3) związane z uczuciami. Odpowiednie do koncepcji Zajonca – korową i wzgórzową [54]. Jak zauważyli Lazarus [52]

oraz Prinz [55], dzięki obecności kory mózgowej w drodze emocjonalnej możliwe jest inicjowanie emocji nie tylko w oparciu o bodźce ze świata zewnętrznego, ale także w oparciu o stany wewnętrzne takie jak przemyślenia lub wspomnienia.



Rysunek 1.3. *Dwie drogi emocji wyróżnione przez LeDoux, na fioletowo zaznaczono drogę korową, a na pomarańczowo drogę wzgórzową,*
źródło: opracowanie własne na podstawie [54]

Ponadto spór dotyczący teorii emocji dotykał także innej ważnej kwestii, a mianowicie uniwersalności i kulturowej niezależności uczuć [50]. Odpowiedzi na pytania dotyczące tych zagadnień dostarczył Ekman badając niepiśmiennych członków plemienia Fore, którzy nigdy nie mieli styczności z kulturą Zachodu. Jego badania nad ekspresjami mimicznymi wykazały, że potrafią oni rozpoznać emocje ludzi pochodzących z innych zakątków Ziemi (rys. 1.4) i że wyraz ich twarzy może być z łatwością odczytywany przez mieszkańców odległych krajów [56]. Pozwoliło to stworzyć koncept, według którego istnieje pewne grono emocji podstawowych. Mają być one charakterystyczne dla każdego przedstawiciela gatunku ludzkiego, niezależnie od jego pochodzenia i wychowania.

Co więcej Ekman w swoich pierwotnych pracach wykazał, że w skład emocji elementarnych wchodzi dokładnie sześć stanów: radość, smutek, zaskoczenie, złość, strach i wstręt. Później nieco wycofał się z tej arbitralnej liczby, sugerując, że liczba bazowych uczuć może być większa [56]. Trochę inaczej do zagadnienia niezależności kulturowej w kwestii emocji podszedł Plutchnik, który wydzielił cztery pierwszorzędowe przeciwstawne pary tych stanów. W jego rozumieniu są one dostępne dla wszystkich ludzi, bo mają podłoże ewolucyjne [57]. Inaczej

sprawa ma się z emocjami wyższymi, które powstają z tych niższych w wyniku uczenia i doświadczania i mogą być niezrozumiałe i obce w odrębnych kulturach.

Rysunek 1.4. *Wyrazy twarzy związane z emocjami podstawowymi*, źródło: [56]

Aspekt społeczny jest szczególnie widoczny w języku, który według najnowszych badań definiuje sposób w jaki postrzegamy rzeczywistość [58]. Jako przykład można tu podać słowo Schadenfreude, określające emocję przyjemności, która towarzyszy człowiekowi, gdy inni odczuwają nieszczęście [56]. Jest on charakterystyczny tylko i wyłącznie dla języków germańskich, a nieobecny we wszystkich innych.

1.3. Programowanie afektywne

Pomimo pewnych ustaleń dotyczących natury uczuć, współczesnej psychologii w dalszym ciągu brakuje jednoznacznej teorii emocji. Nie przeszkadza to jednak w wykorzystywaniu zdobytej wiedzy w celu rozwoju innych gałęzi wiedzy. Taki wniosek towarzyszył wielu naukowcom, a przede wszystkim specjalistom starającym się wykorzystywać interdyscyplinarne podejście do badań. Jedną z nich jest Rosalind Picard, która w 1997 zaproponowała paradygmat informatyki afektywnej [9]. Po raz pierwszy tak wyraźnie zaznaczono potrzebę integracji procesów emocjonalnych w kontaktach z maszynami.

Głównym obszarem zainteresowania programowania afektywnego jest taki rozwój aplikacji, aby możliwe było korzystanie ze stanów emocjonalnych, które towarzyszą ludziom w trakcie obcowania z komputerami. Istotne jest, aby już w fazie projektowania systemów uwzględnić

specyfikę związaną z uczuciami [9, 59] tak, by później móc odpowiednio dostosowywać elementy programów, czy wręcz cały interfejs komunikacyjny.

Wykorzystanie emocji może zatem pozwolić na stworzenie układów, które w oparciu o kontekst mogą dostosowywać swoje działanie i umożliwiać różne sposoby interakcji z nimi. Założenia informatyki afektywnej [9, 21, 60] przejawiają się między innymi w:

1. Zbieraniu danych na temat emocji.
2. Interpretacji skolekcjonowanych informacji.
3. Tworzeniu modeli poszczególnych emocji w celu wyróżnienia oddzielnych uczuć.
4. Tworzeniu jednostkowych modeli afektywnych.
5. Dostosowaniu systemów do aktualnego kontekstu emocjonalnego.
6. Wywoływaniu pożądanych stanów emocjonalnych.

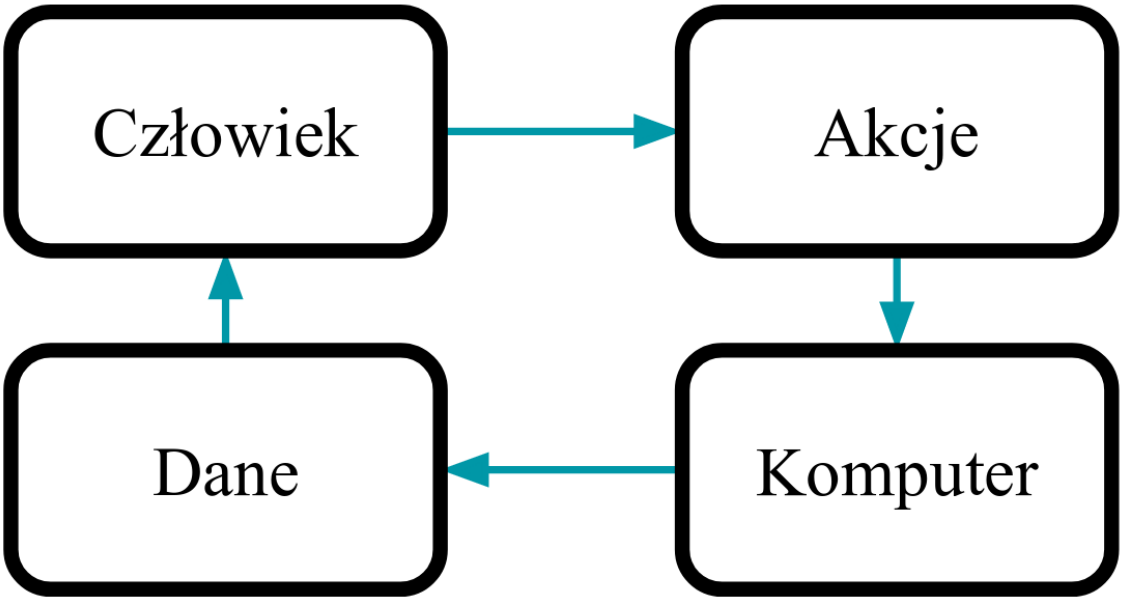
1.4. Pętla afektywna

Realizacja założeń [61] informatyki afektywnej powinna odbywać się z wykorzystaniem tak zwanej pętli afektywnej (rys. 1.5). Jest to znany z dziedziny Automatyki i Robotyki koncept, w którym poprzez wykorzystanie informacji ze środowiska możliwy jest wpływ na system w celu osiągnięcia pożądanego stanu [61].

Przede wszystkim istotne jest wyłuskanie informacji na temat stanu jednostki. Jak wspomniano wcześniej, nie jest to proste zadanie. Szczególnie dlatego, że nie ma zgodności, co do tego, jaka teoria emocji jest najbliższa prawdy. Współcześni naukowcy z zakresu programowania afektywnego w swoich badaniach czerpią więc z różnych koncepcji [21, 62], zgadzając się co do tego, że każda emocja koreluje ze zmianami fizjologicznymi. Jest to istotne założenie z punktu widzenia informatyki, gdyż umożliwia stworzenie i wykorzystanie sensorów, które mogą mierzyć odpowiednie modalności.

Po uzyskaniu informacji na temat obrazu fizjologicznego danej osoby, konieczne jest opracowanie rozwiązań, które umożliwiłyby dekodowanie jej stanów emocjonalnych. Istotny okazuje się tutaj powszechnie stosowany [21, 22, 63] model Russella [64] związany z walencją, która określa charakter – pozytywny bądź negatywny – emocji oraz pobudzenie, które definiuje intensywność danego stanu. Wiążą się one bezpośrednio z reakcjami fizjologicznymi organizmu, przez co rozpoznawanie emocji jest ułatwione. Praktyka pokazuje, że duże znaczenie mają tutaj metody sztucznej inteligencji.

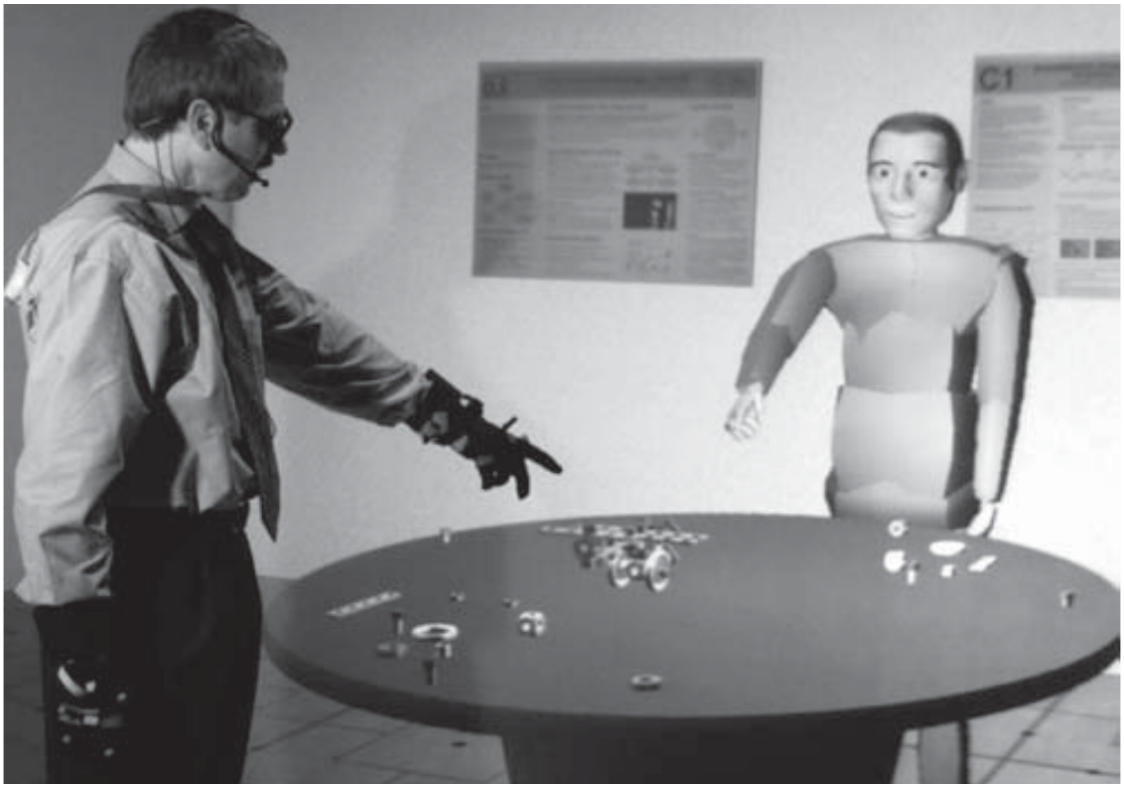
Ważna jest także aplikowalność pętli afektywnej w kontekście niezależności kulturowej. Jak wspomniano wyżej, sugerował to już Ekman [56] poprzez analizę emocji podstawowych i ludzkiej mimiki. Na podstawie literatury, Nogueira wyciągnął analogiczny wniosek na temat sygnałów fizjologicznych [65]. Wykazał, że takie same emocje powodują takie same zmiany w odpowiedziach ciała każdej przebadanej jednostki. Sugeruje to, że możliwa jest generalizacja i tworzenie oprogramowania wykorzystującego pętlę afektywną, która działa analogicznie niezależnie od kultury i okoliczności, w których człowiek dorastał.



Rysunek 1.5. *Cztery elementy pętli afektywnej*,
źródło: opracowanie własne na podstawie [9]

Wyłuskanie stanu emocjonalnego pozwala na taką zmianę systemu, która umożliwia utrzymanie danej sytuacji, bądź generację zmian w celu uzyskania zadanej wartości [66]. W badaniach indukcja stanów emocjonalnych odbywa się przy pomocy bodźców wzrokowych, poprzez prezentację zdjęć ze specjalnie przygotowanej bazy danych [67], z użyciem tekstu, którym jest najczęściej utwór literacki [68], pojedyncze słowa z danego języka [69], bądź dzięki mowie i dźwiękom. W kontekście obrazów szczególne znaczenie mają wyrazy twarzy [53] ze względu na silnie ewolucyjne neurobiologiczne podłoże, predestynujące ludzi do łatwego rozpoznawania ekspresji mimicznych [24, 56, 57].

Jako przykład można tu podać afektywny odtwarzacz muzyki [70], który jest w stanie wpływać na emocje w oparciu o aktualny stan systemu. Wykorzystuje do tego celu dobór odpowiednich piosenek z wcześniej spreparowanej bazy danych.



Rysunek 1.6. *Nauka procedur konstrukcyjnych z wykorzystaniem pomocy Maxa*, źródło: [71]

Można również spotkać się z badaniami opisującymi rolę systemów rekomendujących, które na podstawie zadanych parametrów, między innymi aktualnego nastroju, sugerują podjęcie określonej czynności. Może nią być na przykład obejrzenie konkretnego filmu [72, 73]. Układy dostosowujące się do emocji są także opracowywane w kontekście wirtualnych asystentów (rys. 1.6) takich jak Max [71]. Stwarza to drogę do pokonania barier komunikacyjnych pomiędzy człowiekiem a komputerem.

1.5. Gry afektywne

Ostatnie lata przynoszą wzrost zainteresowania sektorem gier komputerowych w odniesieniu do informatyki afektywnej [63]. Jest to naturalne, gdyż głównym celem tworzenia tych aplikacji jest dostarczenie użytkownikom rozrywki poprzez zapewnienie szeregu wrażeń emocjonalnych [74]. Oceniając jakość zabawy, odbiorcy skupiają się na wielu aspektach, rozpoczynając od wrażeń wizualnych poprzez mechaniki oraz fabułę (rys. 1.7), która towarzyszy im podczas godzin spędzonych przed wyświetlaczem [60]. Wszystkie te charakterystyki składają się na końcowy odbiór produktu.

Zapewnienie odpowiedniego zaangażowania ze strony gracza, czyli dostarczenie rozwiązania, które jest immersyjne [75], jest zadaniem niezwykle złożonym. Aby tego dokonać twórcy gier nie mogą ograniczyć się jedynie do stworzenia interesującej fabuły i intrygującego klimatu. Ważne jest korzystanie z najnowszych osiągnięć informatycznych, które ułatwiają użytkownikom komunikowanie się ze światem gry.

Rysunek 1.7. *Dylematy moralne w Wiedźminie 3, powinności obrońcy ludu kontra inteligentne potwory*, źródło: [76]

Coraz śmielej do tworzenia gier komputerowych używane są okulary wirtualnej rzeczywistości. Pozwalają one pełniej odczuć świat gry, szczególnie w przypadku kiedy aplikacja wymaga kontaktu z innymi osobami. Jako przykład może posłużyć tutaj stworzenie nowych wersji wcześniej wydanych gier jak w przypadku Minecraft, czy The Elder Scrolls V: Skyrim, które jest dostępne na platformie Valve Steam [77]. Z informatycznego punktu widzenia dodanie obsługi nowego systemu sterowania jest zasadniczo prostym zadaniem, co skutkuje pojawianiem się wielu fanowskich modyfikacji oryginalnych tytułów [78]. Zapewniają one wyższej jakości tekstury tak, by obcowanie ze światem wirtualnym było jak najrealniejszym doświadczeniem.

Wykorzystanie emocji wydaje się być sensownym następnym dużym krokiem w branży gier komputerowych. Obecnie gry afektywne pozostają w świetle rozwiązań akademickich i mimo wielu publikacji z tego zakresu wciąż brakuje rozwiązań komercyjnych wywodzących się z tej gałęzi wiedzy [63, 79]. Mimo wszystko, badacze nie szczędzą starań, aby rozwijać dziedzinę, co można pokazać wyróżniając kilka obszarów zainteresowań naukowców.

Na uwagę zasługuje między innymi koncepcja wykorzystania urządzeń służących do grania takich jak pady [80], tworzenie nowych rozwiązań, które dzięki odpowiedniemu wykonaniu nie przeszkadzają w grze [81] i mogą ułatwić (rys. 1.8) sterowanie postacią [82] oraz projektowanie aplikacji do monitorowania stanu użytkownika i zbierania danych [81, 83, 84].

Rysunek 1.8. *Sterowanie robotem za pomocą pasów przyczepionych do rąk*, źródło: [82]

Z punktu widzenia gier bardzo istotne jest operowanie bohaterami niezależnymi tak, aby dostosować ich zachowanie do poziomu emocjonalnego gracza [85], jego doświadczenia i wyników [86] oraz do poziomu wiedzy o świecie gry [87]. Warto tu też wspomnieć o próbie dopasowywania efektów audiowizualnych w grach wyścigowych poprzez zmniejszanie widoczności [88] lub używanie muzyki w celu zwiększenia indukcji strachu w horrorach [89].

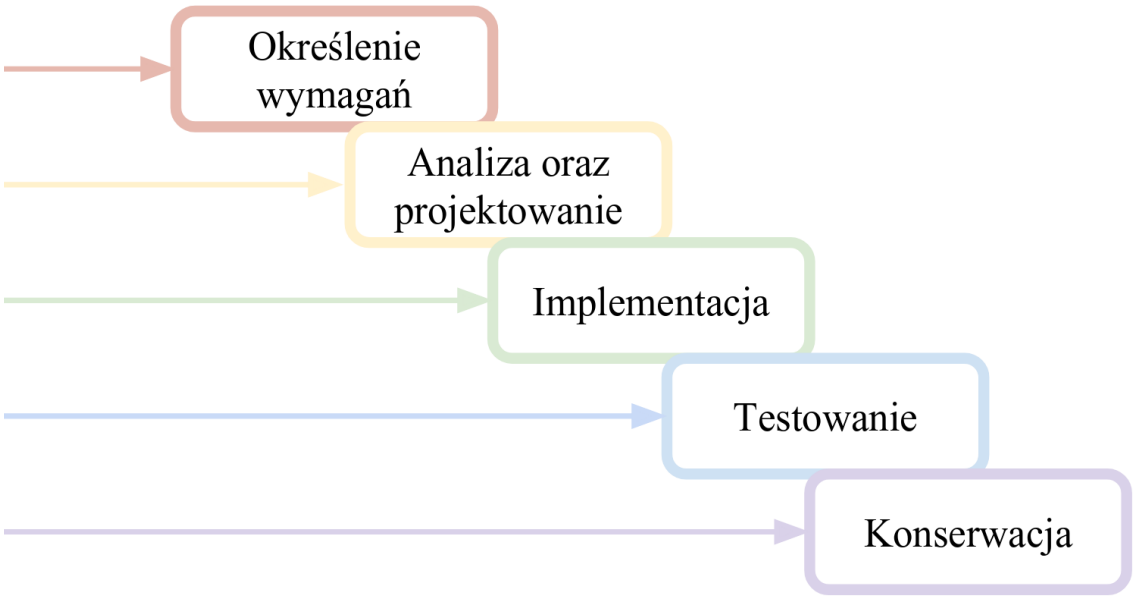
Co więcej, rozwój gier afektywnych widać też na przykładzie dostosowania poziomu trudności gry. Może się to odbywać przez proceduralną generację poziomów jak w przypadku Infinite Mario Bros [90] bądź poprzez adaptację elementów zdolnych do bezpośredniej interakcji z graczem [91], także w grach poważnych [92]. Z punktu widzenia testowania oprogramowania interesujące jest również wykorzystanie pomiaru zachowania użytkowników w celu wychwycenia nieprawidłowości w grach [93].

Jak wskazują poprzednie akapity, postęp w dziedzinie gier afektywnych dotyczy wielu zróżnicowanych elementów. W obecnej literaturze istnieje jednak niewiele pozycji, które skupiałaby się bezpośrednio na procesie projektowania gier [60, 63] jako aplikacji wywołujących emocje. Jest to istotne niedopatrzenie ze względu na to, że to właśnie faza konceptualna niesie za sobą najpoważniejsze implikacje w kontekście wytwarzania oprogramowania [94]. To właśnie wtedy czynione są założenia dotyczące przyszłych systemów i to właśnie wtedy należy próbować przewidzieć jakie cele oraz w jaki sposób mogą zostać osiągnięte.

2. Gry komputerowe

2.1. Proces tworzenia

Jak każdy dobrze wykonany system informatyczny, gry komputerowe czerpią garściami z inżynierii oprogramowania. Definiuje ona kilka kluczowych etapów (rys. 2.1), przez które twórcy aplikacji muszą przejść, aby osiągnąć zamierzony efekt [95]. Są to między innymi: rozpoznanie rynku i odbiorców programu, analiza celów i właściwości projektowanego systemu, definiowanie projektu i jego implementacja oraz testowanie i zatwierdzanie spełnienia kryteriów, a także dostosowywanie w ramach ewolucji względem zmieniających się oczekiwań.



Rysunek 2.1. *Jeden z modeli, przedstawiający poszczególne etapy tworzenia oprogramowania*, źródło: [95]

Jak każdy dobrze wykonany współczesny system informatyczny, gry komputerowe czerpią garściami z dziedzin potencjalnie wydawałoby się niezwiązanych z oprogramowaniem [7]. Mowa tutaj na przykład o psychologii, socjologii, czy szeroko pojętej sztuce. Co więcej, gry

wideo w samej swojej istocie są bytem interdyscyplinarnym. Ich powstanie jest wielkim przedsięwzięciem, angażującym rzesze ludzi i pochłaniającym ogromne nakłady finansowe [22].

Aby mówić o grze komputerowej czasem wystarczy przenieść interakcje znane z gier planszowych lub zgadywanek słownych. Potrzebne są cel, zasady oraz interfejs umożliwiający oddziaływanie na program. Dotyczy to oczywiście jedynie najprostszych projektów. W przypadku bardziej skomplikowanych systemów można mówić o dodatkowych elementach takich jak opracowanie scenariusza, projektu graficznego i oprawy dźwiękowej. Ważne jest także zespolenie tych elementów w spójną całość, tworzącą niepowtarzalny klimat [44, 96].

Z perspektywy gier afektywnych istotne jest również skupienie się na czynnikach dotyczących emocji. Jak uwypuklono we wcześniejszym rozdziale wymagania związane z tym sposobem programowania dotyczą w równej mierze reprezentowania i wywoływania oraz odczytywania stanów emocjonalnych w taki sposób, aby umożliwić stworzenie pętli afektywnej.

Hudlicka pokazuje, ze już na wczesnym etapie projektu gry komputerowej, działającej w oparciu o uczucia, twórcy muszą sobie odpowiedzieć na pewne pytania dotyczące jej struktury [60, 85]. Koncentrują się one na rodzaju wykorzystywanych emocji, zmianach jakie będą wywoływane w aplikacji, czy poziomie wymaganego realizmu.

Z pomocą przychodzi tutaj opracowanie stworzone przez Gilleade, Dix oraz Allanson [97]. Wskazują oni trzy podstawowe heurystyki ułatwiające proces projektowania:

1. „Stwórz przede mną wyzwanie.”
2. „Pomóż mi.”
3. „Wywołaj we mnie emocje.”

Pierwsza z nich dotyczy sytuacji, w której przed graczem stawiane są kolejne utrudnienia. Odpowiednie przewidzenie takich sytuacji w procesie projektowania gier, może pozytywnie wpłynąć na zaangażowanie odbiorców. Już od dawna dostosowanie poziomu trudności jest wykorzystywane w komercyjnych grach komputerowych. Najczęściej wygląda to tak, że użytkownik arbitralnie ustala stopień skomplikowania na początku rozgrywki lub w dowolnym późniejszym momencie. Wykorzystanie paradygmatu afektywnego może pomóc zautomatyzować ten proces, zmniejszając, a docelowo eliminując, znudzenie gracza. Heurystyka przeznaczona jest dla zaawansowanych odbiorców, którzy od początku dysponują pewnymi umiejętnościami.

Kolejna heurystyka korzysta z poziomu frustracji użytkownika i jest przeznaczona przede wszystkim dla graczy, którzy nie są profesjonalistami. Zwraca ona uwagę projektanta na sytuacje, w których odbiorca jest na tyle zirytowany, że rozważa zakończenie rozgrywki. Oferuje podjęcie działań w postaci obniżenia poziomu trudności, podpowiedzi pojawiających się na ekranie (rys. 2.2) i wskazówek udzielanych przez bohaterów niezależnych. Analogicznie do

pierwszej sytuacji, heurystyka pomocy przeznaczona jest przede wszystkim dla niezaawansowanych graczy, by zaangażować ich w produkt.

Rysunek 2.2. *Podpowiedzi umożliwiające lokalizację celu w Assassin's Creed 3*, źródło: [98]

Ostatni sposób projektowania gier afektywnych ma na celu wywołanie specyficznych reakcji emocjonalnych u odbiorców. Heurystyka ta jest najmniej ścisła i dotyczy szerokiej liczby zagadnień. Hudlicka [60] sugeruje, że korzystanie z niej wiąże się z opracowaniem modeli poszczególnych emocji, co również jest istotne z perspektywy projektu gry.

2.2. Narzędzia wspierające budowanie

Jak wskazano wyżej, tworzenie gier komputerowych jest bardzo złożonym procesem. Wymaga pracy autorów z różnorodnych dziedzin naukowych, angażując setki ludzi w celu stworzenia wielkich produkcji. Wyróżniające zadanie mają tutaj twórcy oprogramowania, od których zależy ostateczny wygląd rozgrywki, jej optymalizacja i wierność odwzorowania wszystkich detali stworzonego świata.

Każde uniwersum jest tworem wieloaspektowym, które współcześnie opisuje się przy pomocy wzorów matematycznych i fizycznych (rys. 2.3). W zależności od stopnia złożoności świata, w którym osadzona jest gra komputerowa, różna liczba czynników musi zostać uwzględnionych [99]. Mowa tutaj na przykład sposobie w jaki poruszają się bohaterowie, jak

zachowuje się broń i naboje, a także o bardziej zaawansowanych fenomenach jak grawitacja, pogoda i zjawiska z nią związane.

Wiele współczesnych gier umożliwia wieloosobową rozgrywkę, wykorzystując do tego połączenie internetowe. Użytkownicy współpracują ze sobą w celu osiągnięcia konkretnego celu. Prowadzone są również turnieje sportowe z dużymi nagrodami [100]. Od strony programistycznej wymaga to obsługi wielowątkowości oraz opracowania rozwiązań umożliwiających synchronizację wielu klientów z najwyższą dokładnością.

Rysunek 2.3. *Fizyka pocisku, która stanowi jednocześnie jedną z najważniejszych mechanik w serii Angry Birds*, źródło: [101]

Złożoność interakcji, które muszą zostać opracowane przy pomocy oprogramowania, skutkuje tym, że w kontekście gier komputerowych mówi się o silnikach [102]. Umożliwiają one obsługę grafiki, wejścia, wyjścia, metod sztucznej inteligencji, fizyki, wykrywania zderzeń między elementami gry oraz wspomnianej wyżej sieci. Ponadto, każdy silnik związany jest z odpowiednią platformą, która dostarcza rozwiązania do tworzenia kodu, a często opracowywania elementów grafiki i dźwięku.

Jak pokazał Ahmad [103] stworzenie narzędzia, które umożliwiłoby tworzenie gier dowolnego gatunku jest praktycznie niemożliwe. W związku z tym największe firmy dysponują swoimi rozwiązaniami [103], które są udoskonalane wraz z kolejnymi premierami. Przykładem może być Anvil wykorzystany w kolejnych odsłonach Assassin's Creed bądź REDEngine użyty do produkcji gier z serii Wiedźmin.

Wspomniane wyżej technologie są najczęściej wykorzystywane tylko w obrębie danej spółki i niedostępne w domenie publicznej. Największa część publikacji naukowych dotyczy zatem rozwiązań otwartych, które omówiono poniżej. Wyróżniono takie kwestie jak stabilność, dostępność dokumentacji, wsparcie środowiska, popularność, platformy oraz języki, w którym można tworzyć aplikacje.

2.2.1. CryENGINE

CryENGINE [104] umożliwia projektowanie oprogramowania tylko z użyciem systemu Windows. Całość została napisana przy pomocy języków C++, Lua oraz C# i również w takich technologiach umożliwia tworzenie aplikacji. Mogą być one budowane na różne platformy, w tym Windows oraz Linux, a także na różne generacje konsol Xbox, PlayStation oraz Wii. Do najbardziej znanych gier opracowanych przy pomocy tego silnika można zaliczyć Crysis (rys. 2.4), Far Cry i Sniper Ghost Warrior.

Rysunek 2.4. *Fotorealistyczne możliwości silnika CryENGINE na przykładzie gry Crysis 3*, źródło: [105]

Tworzenie gier komputerowych odbywa się poprzez dodawanie elementów o zdefiniowanych parametrach do pustego pierwotnie świata [104]. W założeniu twórców obecność atrybutów ma ułatwić projektowanie dużych przestrzeni poprzez zaimplementowane algorytmy automatycznej generacji świata. Dużym wyróżnikiem platformy CryENGINE jest możliwość szybkiego przełączania się miedzy widokiem kodu a obecnym wyglądem gry, co osiągnięto uruchomieniem silnika gry wraz z edytorem.

Największą zaletą środowiska są niewątpliwie zaawansowane możliwości graficzne, które były dostępne już w pierwszej wersji silnika. Warto wyróżnić też to, że oprogramowanie pozostaje darmowe w kontekście tworzenia gier niezależnych oraz dla środowiska akademickiego [104]. Z wad należy podkreślić słabą optymalizację produktu oraz okrojoną dokumentację znajdującą się na stronie producenta. CryENGINE zostaje często wskazywany jako najtrudniejszy i najbardziej wymagający spośród ogólnodostępnych silników gier elektronicznych [106].

2.2.2. Game Editor

Game Editor [107] należy do najprostszych i najmniej zaawansowanych silników gier komputerowych. Jest dostępny na system Windows, Linux oraz macOS, umożliwiając budowanie aplikacji na wskazane platformy oraz urządzenia napędzane przez iOS i Windows 10 Mobile. Korzysta w całości z niskopoziomowego języka podobnego do C, co skutkuje niską popularnością rozwiązania wśród współczesnych programistów.

Koncept opracowywania aplikacji z wykorzystaniem programu Game Editor sprowadza się do opracowania aktorów oraz obsługi z góry określonych zdarzeń. Ważnym elementem dostępnym poprzez silnik są ścieżki, określające sposób poruszania się obiektów w grze [107].

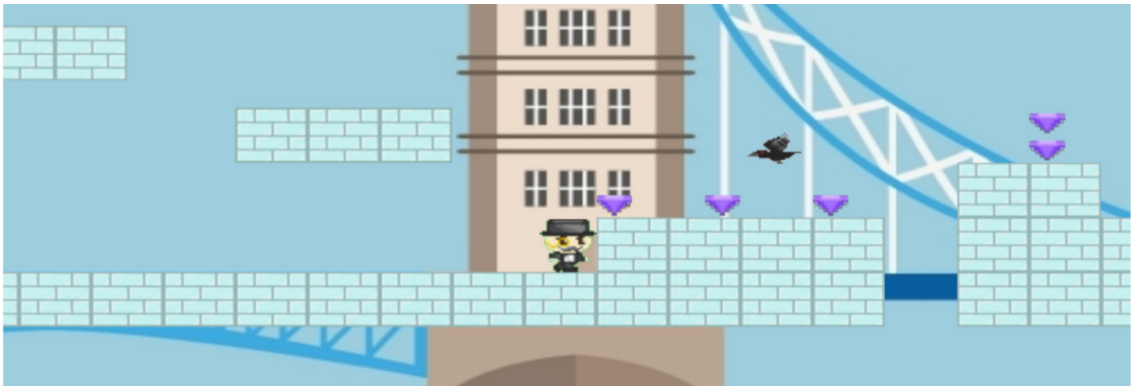
Wśród wad oprogramowania często wymienia się niską stabilność środowiska, widoczną szczególnie przy okazji tworzenia dużych projektów [108]. Co więcej, wskazane problemy mogą uwidocznić się także przy okazji opracowywania poziomów, które zaleca się rozbijać na mniejsze części [107]. Opisane niedociągnięcia oraz brak wsparcia dla gier trójwymiarowych skutkują tym, że Game Editor mimo prostoty środowiska, nie zyskał popularności.

2.2.3. GameMaker

Możliwość tworzenia oprogramowania przez zaawansowanych programistów i laików zapewniła GameMakerowi znaczną popularność [109]. Narzędzie jest dostępne na systemy Windows i macOS oraz umożliwia opracowywanie rozwiązań zarówno na te platformy (rys. 2.5), ale też na środowiska mobilne i webowe. Językiem programowania dostępnym na platformie jest Game Maker Language bazujący na składni wprowadzonej przez C++ oraz JavaScript [110].

Gry komputerowe z wykorzystaniem GameMakera tworzone są w oparciu o mechanizm przeciągania i upuszczania elementów [110]. Każdy obiekt aplikacji ma następnie określane standardowe parametry i może zostać związany z dodatkowym skryptem.

Niewątpliwą zaletą silnika jest wysoka stabilność oraz możliwość korzystania z rozszerzeń w postaci bibliotek oferowanych przez twórców i rozbudowaną społeczność. Wadą natomiast może być ograniczone zainteresowanie profesjonalistów. Skutkuje to tym, że z wykorzystaniem

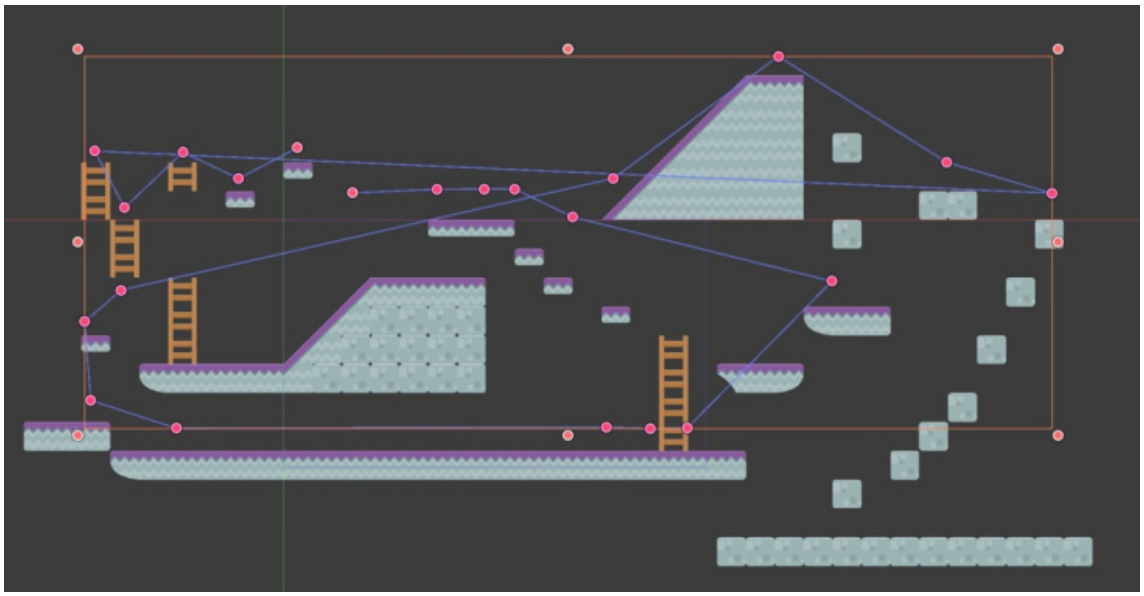


Rysunek 2.5. *London Bridge wykonane z użyciem GameMakera*, źródło: [21]

GameMakera powstają stosunkowo proste produkcje [109], a samo środowisko wykorzystywane jest zazwyczaj jako pierwszy element na drodze do nauki tworzenia gier komputerowych.

2.2.4. Godot

Podobnie jak GameMaker, Godot umożliwia tworzenie aplikacji na wszystkie najpopularniejsze środowiska [111]. Edytor tego silnika dostępny jest na system Windows, macOS oraz najbardziej rozpowszechnione dystrybucje oparte o jądro Linux. Całość napisana została przy pomocy języków C oraz C++, a gry oparte na tym oprogramowaniu mogą być budowane z użyciem języka C++ oraz własnego wysokopoziomowego rozwiązania o nazwie GDScript.



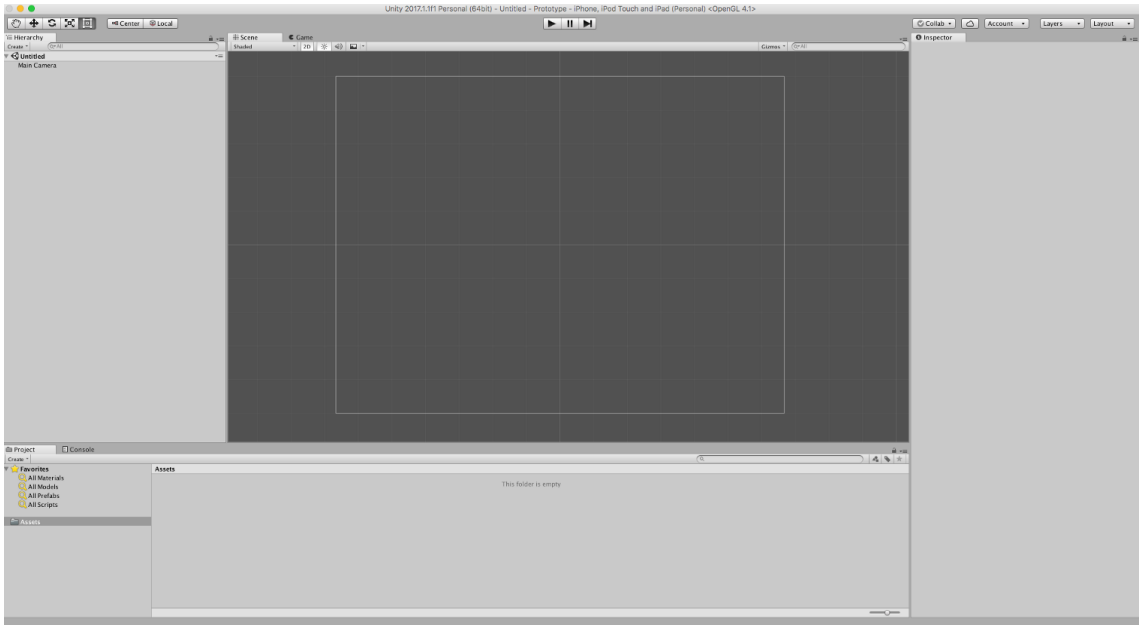
Rysunek 2.6. *Powiązania pomiędzy scenami w poziomie dla silnika Godot*, źródło: [112]

Tworzenie gier komputerowych przy użyciu programu Godot polega na projektowaniu poszczególnych scen, z których przygotowywany jest graf (rys. 2.6) danego poziomu [111]. Narzędzie bardzo silnie stawia na współpracę programistów pracujących w ramach technik kontroli wersji. Wszystkie elementy, używane w ramach procesu tworzenia gier, są zapisywane nie w bazie danych, ale z wykorzystaniem systemu plików.

Niewątpliwą zaletą silnika jest wspieranie opracowywania wszystkich elementów związanych z grą, takich jak grafika oraz ścieżka dźwiękowa. Na pochwałę zasługuje też dostępność dużej liczby materiałów umożliwiających naukę tworzenia gier z wykorzystaniem silnika. Nie przekłada się to niestety na duże zainteresowanie społeczności twórców [113].

2.2.5. Unity

Przy pomocy Unity można tworzyć zarówno gry dwuwymiarowe jak i trójwymiarowe. Za wielką popularnością tego silnika, podobnie jak w przypadku GameMakera, przemawiają dwa rodzaje interfejsów [106, 114]. Pierwszy przeznaczony jest dla początkujących użytkowników, drugi zaś dla profesjonalnych programistów.



Rysunek 2.7. *Interfejs programistyczny dla silnika Unity*, źródło: [115]

Zaawansowane skrypty pisane są przy pomocy języka C#. Środowisko uruchomieniowe silnika zostało napisane przy pomocy technologii C++. Unity pozwala na tworzenie oprogramowania na wszystkie popularne systemy operacyjne komputerów, telefonów, konsol, okularów wirtualnej rzeczywistości, a nawet przystawek telewizyjnych i telewizorów [114]. Najbardziej znane gry stworzone przy pomocy tego silnika to Hearthstone oraz Gwint.

Tworzenie gry z wykorzystaniem platformy Unity podzielone jest na kilka etapów. Z perspektywy projektanta ważne jest rozróżnienie pomiędzy elementami interfejsu graficznego. Wyraźnie zaznaczono informacje na temat projektu, jego hierarchii, konsolę, informacje na temat zaznaczonego obiektu i dwa widoki – aktualną scenę oraz obraz z perspektywy gracza. W kreowanym świecie gry osadza się obiekty, którym przypisuje się odpowiednie parametry oraz skrypty, odpowiadające między innymi za ich interakcję z graczem bądź ruch [114].

Unity to najpopularniejsze dostępne powszechnie narzędzie do tworzenia gier [106]. Przejawem tego jest ogromna ilość materiałów dostępnych w specjalnie przygotowanym sklepie internetowym, rozbudowana dokumentacja techniczna oraz mnogość poradników zarówno dla początkujących jak i zaawansowanych. Warto podkreślić też niskie wymagania sprzętowe oprogramowania oraz dostępność darmowej wersji [114]. Minusy związane są z dużymi projektami. Dotyczą częściowej awaryjności narzędzia oraz braku rozbudowanych funkcji.

2.2.6. Unreal Engine

Unreal Engine to silnik, który do niedawna był kojarzony przede wszystkim ze studiami produkującymi największe gry komputerowe [116]. Jego czwarta wersja przyniosła zmianę, gdyż niezależnym deweloperom udostępniono odpłatną wersję oprogramowania. Jest to najbardziej rozbudowane z rozwiązań, które opisano. Ustępuje jedynie innym silnikom wykorzystywanym przez największe studia projektowe. Całość napisano w języku C++ i to właśnie w tym języku tworzy się gry komputerowe z wykorzystaniem Unreal Engine [116]. Z pomocą tego rozwiązania zaprojektowano takie gry jak BioShock (rys. 2.8) i Gears of War.

Rysunek 2.8. *Walka w jednej z gier z serii BioShock*, źródło: [117]

Aby stworzyć grę z wykorzystaniem tego narzędzia można posługiwać się systemem powiązań, które opisują możliwe scenariusze występujące w grze oraz interakcje pomiędzy pojedynczymi obiektami, czyli tak zwaną logikę gry, bez konieczności pisania kodu. Można również nie sięgać do tych rozwiązań i wykorzystać jedynie rozbudowane narzędzia programistyczne [116].

Atutem, który wyróżnia Unreal Engine na tle innych silników jest dostęp do jego kodu źródłowego [106, 116]. Pozwala to na modyfikację i dostosowanie oprogramowania do własnych potrzeb, wymagając jednak specjalistycznej wiedzy programistycznej. Użycie języka C++ można potraktować zarówno jako wadę, ze względu na to, że wymaga większego czasu na opanowanie, ale także jako zaletę, gdyż gry pisane przy jego pomocy są znacznie szybsze niż w przypadku metod wysokopoziomowych [116]. Warto zwrócić uwagę na ilość materiałów, złożoność dokumentacji oraz wsparcie społeczności, które mimo że znacznie mniejsze niż w przypadku Unity, są istotną pomoc dla początkujących.

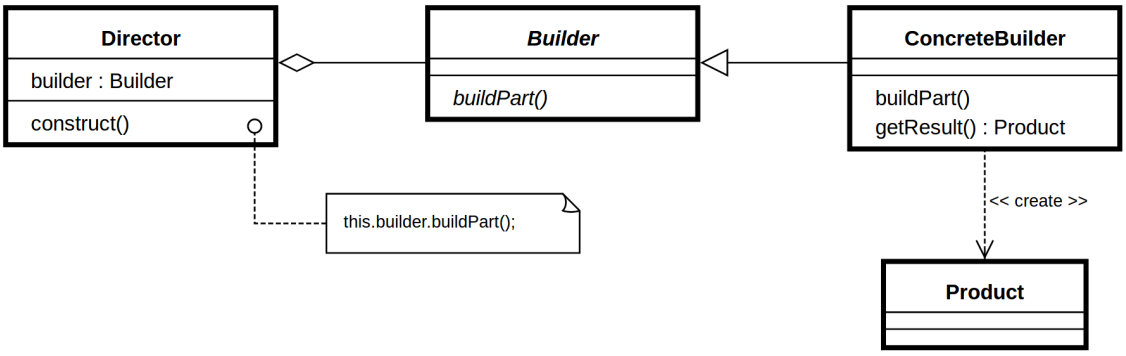
2.3. Growe wzorce projektowe

Wzorce projektowe odgrywają szczególną rolę w kontekście tworzenia oprogramowania ze względu na uniwersalność oraz możliwość łatwego rozwiązywania powtarzających się problemów. Zostały one wyróżnione przez Alexandra [118] już 1977, by w 1995 zostać opublikowane przez autorów identyfikujących się jako „Banda czworga” i stać się jedną z najbardziej wpływowych pozycji w historii informatyki [119]. Zaproponowana przez nich struktura wzorca składa się z trzynastu elementów:

1. Nazwa umożliwiająca identyfikację.
2. Cel oraz przeznaczenie.
3. Inne nazwy.
4. Motywacja, kryjąca się za tym, by go użyć.
5. Sytuacje, kiedy może być użyteczny.
6. Reprezentacja graficzna.
7. Stosowane klasy i obiekty.
8. Interakcja pomiędzy uczestnikami.
9. Konsekwencje płynące z jego użycia.

10. Specyficzne kwestie związane z implementacją.
11. Przykładowy kod.
12. Znane przykłady zastosowania.
13. Odniesienie do pokrewnych wzorców.

Przykładem może być tutaj wzorzec Budowniczy (rys. 2.9), którego podstawowym zadaniem jest podzielenie procesu kreowania obiektu na kilka mniejszych etapów. Efektem jest rozdzielenie tworzenia obiektu od jej reprezentacji [119].



Rysunek 2.9. *Diagram klas dla wzorca Budowniczy*, źródło: [119]

Także gry komputerowe doczekały się swoistej interpretacji wzorców projektowych wykonanej w 2004 roku przez Björka i Holopainena [23]. Wykorzystali oni informacje dotyczące sposobu w jaki projektowane są gry, wskazując na powtarzalność wykorzystywanych schematów, reguł oraz mechanik growych. Ludologia definiuje je jako ograniczenia i zasady dobrowolnie przyjmowane przez uczestników rozgrywki [96]. W ujęciu Björka i Holopainena [23] rola projektanta gry komputerowej może zostać sprowadzona do opracowania takich ram, które umożliwią czerpanie z niej przyjemności [21]. Ważne jest zatem współoddziaływanie schematów oraz określenie konsekwencji płynących z ich zastosowania dla całej rozgrywki. Forma wzorców projektowych używanych w grach ma postać pięcioelementową [23]. Wyróżniono:

1. Nazwę umożliwiającą identyfikację.
2. Cel oraz przeznaczenie.
3. Przykłady z istniejących gier.
4. Konsekwencje płynące z jego użycia.
5. Relacje z pozostałymi wzorcami. Wyszczególniono tutaj:

(a) Inne wzorce, które są instancjonowane przez ten wzorzec.

(b) Inne wzorce, na które wywarty zostaje wpływ przez użycie tego wzorca.

(c) Inne wzorce, które instancjonują dany wzorzec.

(d) Inne wzorce, które wpływają wpływ na dany wzorzec.

(e) Inne wzorce, które mogą kolidować z danym wzorcem.

Bardzo dobrym przykładem growego wzorca projektowego jest Niedoskonała Informacja, której podstawowym zadaniem jest dostarczenie istotnych komunikatów w ograniczonej formie bądź celowe wprowadzanie gracza w błąd. Często spotykany jest on podczas podawania statystyk na zakończenie danego poziomu, jak ma to miejsce w grach DOOM, bądź Assassin's Creed. Użytkownik zostaje powiadomiony o ilości odkrytych sekretów [23], co może mieć generować radość, złość bądź zmiany afektywne [21].

2.4. Afektywne growe wzorce projektowe

Idea wzorców emocjonalnych skojarzonych z grami komputerowymi pojawiła się w zasadzie od razu po opublikowaniu zbiorów wzorców projektowych przez Björka i Holopainena [23]. Jeszcze tego samego roku wydano zestawienie, w którym pokazano, że różne rodzaje gier mogą indukować odrębne stany emocjonalne. Kluczową rolę odgrywały tutaj gatunki. Puzzle złączono z relaksem, a gry akcji miały korelować z gniewem oraz strachem [120]. Badacze wskazali na potencjalne zastosowanie opracowanych wyników w procesie projektowania gier i tworzenia immersyjnych systemów.

Później koncepcję tę rozbudowano wskazując, że stany emocjonalne mogą być wywoływane nie tylko przez całościowy odbiór gry, ale przez jej konkretne elementy [121]. Do podobnych wniosków doszła także wspominana wcześniej Hudlicka [60]. Przykładem zastosowania tego podejścia może być zarówno odpowiednio dobrana grafika, oświetlenie oraz kontrast w grze DOOM jak i tajemnicze zagadki w Myst. Z perspektywy nowszych wydań szczególnie dobrze może to być widoczne po rozbudowanej fabule, kiedy odbiorca utożsamia się z przeciwnościami losu postapokaliptycznej This War of Mine (rys. 2.10), czy współczuje ojcu poszukującemu córki w produkcji Wiedźmin 3: Dziki Gon. Badaczka nazywa to postępowanie mapowaniem bodźców do emocji [60] i zwraca uwagę, że jest to wymagający proces ze względu na to, że stymulacja może skutkować różnymi efektami u różnych osób.

Co ciekawe, Hudlicka wskazała, że już Picard w pierwszej publikacji dotyczącej programowania afektywnego [9] zwracała uwagę na podobny sposób tworzenia gier. Badaczka sugerowała, aby w fazie projektowania związać potencjalne, ukryte stany emocjonalne oraz zachowania odbiorcy z czynnikami, które mogą je wywoływać. Mowa tu między innymi o utracie

punktów lub zasobów w grze komputerowej. Przy pomocy diagramu stanów lub łańcuchów Markowa, użytecznych w kontekście sztucznej inteligencji i określania prawdopodobieństw wystąpienia konkretnych zdarzeń, możliwe miałoby być projektowanie i modelowanie programów afektywnych [9].

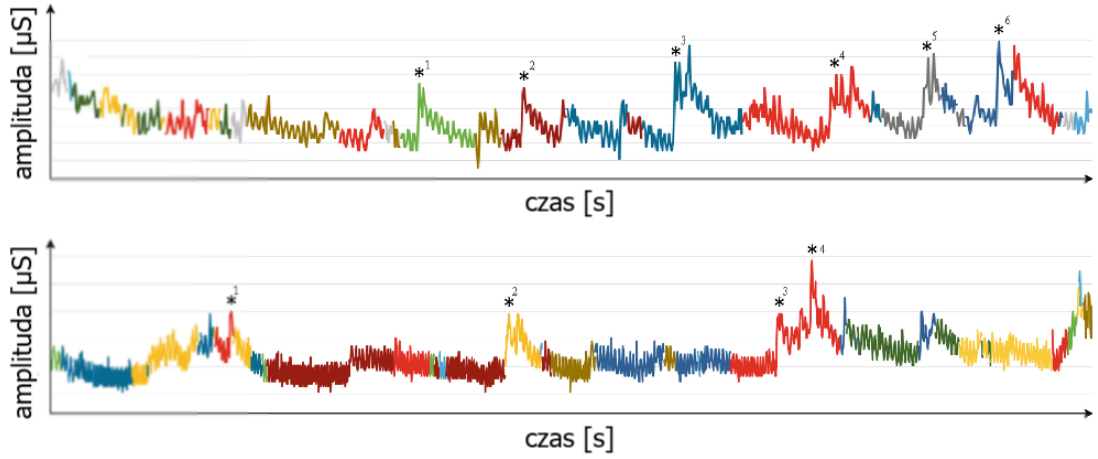
Rysunek 2.10. *Nagła śmierć bohatera w This War of Mine, powodująca znaczne komplikacje w dalszej rozgrywce*, źródło: [122]

Dormann, Whitson i Jennifer wychodząc bezpośrednio od idei Björka i Holopainena [23] przedstawili zestaw wzorców projektowych nakierowanych na afekt [20]. Ich celem było pokazanie oddziaływania gier na rozumienie ważnych kwestii społecznych takich jak inteligencja emocjonalna, troska o środowisko, czy walka ze stereotypami. Każdy wzorzec z opracowanego zestawu przypisano do jednej z trzech grup: rozumienie uczuć, interakcje społeczne oraz nauka w oparciu o stany emocjonalne. Jak sami zaznaczają, nie przedstawili żadnych analiz oraz wniosków w kontekście programowania afektywnego, sugerując, że do tego potrzebne jest opracowanie konkretnych gier komputerowych [20].

W publikacji [21] zebrano pomysły związane z procesem projektowania afektywnych gier komputerowych oraz sformalizowano i ugruntowano ideę afektywnych wzorców projektowych. Badacze ponownie wyszli bezpośrednio od pomysłu Björka i Holopainena [23], wybierając niewielki zestaw wzorców, które mogą powodować pojawienie się odpowiedzi fizjologicznej u osoby uczestniczącej w eksperymencie. Przygotowano dwie wersje gry platformowej w środowisku GameMaker dla grupy kontrolnej i badawczej oraz przedstawiono pewne wstępne wyniki. Nie opracowano jednak złożonej analizy w odpowiedzi na konkretne wzorce. Co więcej,

użyta gra nie wdrażała w pełni ideałów programowania afektywnego ze względu na to, że nie użyto pętli afektywnej.

Tekst [123] stanowi kontynuację poprzednio wymienionej publikacji. Autorzy uzupełnili pomysł związany z afektywnymi growymi wzorcami projektowymi o dalszą analizę. W badaniu wykorzystano grę logiczną o nazwie Crime Secrets: Crimson Lily. Naukowcy przeprowadzili rozbiór gry na wzorce projektowe użyte w trakcie implementacji oprogramowania, a następnie przeprowadzili obserwację, podczas której zaproszeni uczestnicy mieli przez dłuższy czas grać na komputerze. Podczas sesji eksperymentalnej, osoby badane były podłączone do urządzenia rejestrującego sygnały fizjologiczne. Ze zgromadzonych danych stworzono wykresy (rys. 2.11) i zaznaczono miejsca występowania wzorca projektowego. Okazało się, że odpowiedzi na dany bodziec są widoczne na pierwszy rzut oka [123]. Ponownie nie przeprowadzono żadnej głębszej analizy i przebadano tylko wąską grupę ludzi. W związku z tym, że skorzystano z gotowej aplikacji ponownie nie użyto pętli afektywnej.



Rysunek 2.11. *Rozkład reakcji elektrodermalnej podczas gry z zarejestrowanymi miejscami wystąpień growych wzorców projektowych*, źródło: [123]

Kolejna publikacja [22] z zakresu afektywnych wzorców projektowych w grach komputerowych korzysta z własnego rozwiązania implementującego pętlę afektywną. Grę stworzono w środowisku Unity. Mimo pewnych nieścisłości w nazewnictwie, także w tym opracowaniu korzystano z growych wzorców projektowych Björka i Holopainena [23] ograniczając się jednak tylko do rozpoznania kilku mechanik afektywnych. Nie dostrzeżono w ten sposób wielu wysokopoziomowych modeli, które mogą mieć znaczenie afektywne. Mowa tutaj na przykład o walce z przeciwnikami, czy poznawaniu fabuły gry. Autorzy pokusili się o częściową analizę,

zaprezentowali jednak wykresy z rozgrywki części uczestników badania, nie sporządzono natomiast interpretacji przekrojowych w odniesieniu do konkretnych wzorców u poszczególnych badanych. W tekście padło także kilka mocnych stwierdzeń dotyczących kierunku korelacji bez uzasadnień statystycznych. Mimo niezaprzeczalnego wkładu w dziedzinę programowania afektywnego i sposobu projektowania gier, w badaniu [22] skorzystano z urządzeń klasy laboratoryjnej. Jak wskazują naukowcy wspomniani wyżej, takie eksperymenty są istotne z punktu widzenia akademickiego [21], ale mają nikłą szansę wpływu na środowisko gier komercyjnych.

Z przeprowadzonych badań literaturowych jasno wynika, że w kontekście gier afektywnych wciąż brakuje publikacji, która korzystając z growych wzorców projektowych w procesie projektowania aplikacji, implementowałaby pętlę afektywną. Istotne jest stworzenie rozwiązania, opierającego się przy tym na urządzeniach nasobnych, które nie są tak ograniczające jak sprzęt klasy laboratoryjnej. Poniższa praca stanowi próbę zapełnienia luki w tej części nauki o systemach komputerowych.

3. Kontrolowanie i interpretacja aktywności

3.1. Monitorowanie funkcjonowania

Anatomowie i psychologowie obserwując stany emocjonalne odkryli, że łączą się one ze specyficznymi zmianami w ciele oraz w zachowaniu człowieka. Spostrzeżenie to niewątpliwie wpłynęło na Picard, kiedy publikowała pierwsze prace z zakresu programowania afektywnego [9]. Postęp technologiczny, który dokonał się od tamtego czasu stworzył dogodne warunki do dokładniejszego monitorowania funkcjonowania człowieka, pozwalając na lepszą interpretację uczuć, które mu towarzyszą i w efekcie tworzenie wolnego od błędów oprogramowania afektywnego, co jest istotne z perspektywy inżynierii wymagań [95].

Hudlicka [60, 85], na co zwrócono uwagę już w pierwszym rozdziale i co przywołano powyżej, stwierdziła, że emocje wyrażają się wśród ludzi przez różne modalności (rys. 3.1). Odnosząc to do fazy projektowania gier z rozdziału drugiego, twórca oprogramowania musi zdecydować się, których z nich będzie wykorzystywać tworząc koncept aplikacji afektywnej.

W publikacji [90] zasugerowano podział sposobów mierzenia stanów emocjonalnych w programowaniu afektywnym na:

1. Subiektywne.

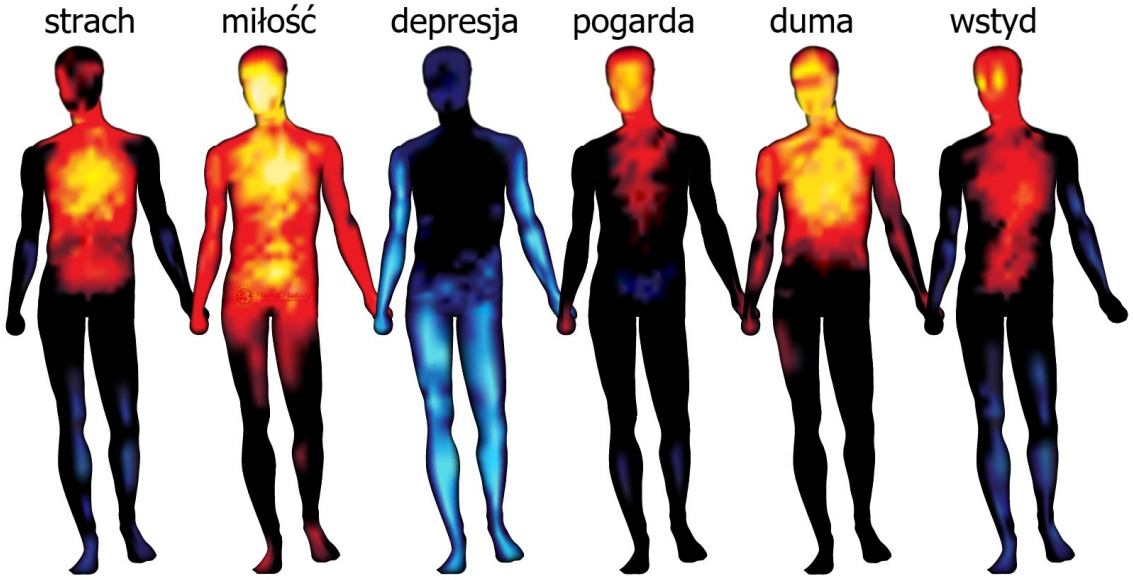
2. Obiektywne.

3. Wywodzące się z aktywności w grach.

Pierwszy odnosi się do sytuacji, w której użytkownik opowiada o swoich przeżyciach związanych z daną sytuacją emocjonalną. Autorzy podziału zwracają uwagę, że metoda ta mimo bezsprzecznej dokładności, jeśli używana w trakcie procesu korzystania z gry, może przeszkadzać w naturalnym odbiorze treści. W przypadku stosowania po rozgrywce może okazać się nietrafna ze względu na problemy z pamięcią u osoby badanej, analogicznie do introspekcji używanej we wczesnej psychologii [124].

Drugi uwzględnia dane fizjologiczne, przy czym w kwestii interpretacji uzyskanych informacji ważny jest też bardziej szczegółowy podział. Jedne z nich wykorzystują modele, np. bazując na metodach sztucznej inteligencji. Inne zaś korzystają z mapowania opartego o komentarze użytkownika. Do zalet podejścia obiektywnego należy relatywnie niewysoka inwazyjność i znacząca skuteczność.

Ostatni wykorzystuje wzorce zachowań wygenerowane w oparciu o proces grania z wykorzystaniem komputera. Mocną stroną tego sposobu jest najmniejsza inwazyjność, bo gracz nie musi być ich w ogóle świadom jak to ma miejsce w przypadku współpracy ze sprzętem [60]. Z drugiej jednak strony projektant oprogramowania afektywnego musi liczyć się z niską rozdzielczością systemu.



Rysunek 3.1. *Temperatura jako jedna z modalności, dzięki którym można rozpoznać emocje*, źródło: [125]

W kontekście aplikowalności rozwiązań afektywnych do systemów komercyjnych, ważne jest aby zwrócić uwagę na cechy indywidualne związane z emocjami. Hudlicka mówi tutaj o afektywnych cechach osobowości. Są to specyficzne dla każdej jednostki zdolności do reagowania na jakąś sytuację [60]. Warto też zwrócić uwagę na czynniki takie jak nastrój oraz poziom umiejętności i wprawę danej osoby w graniu z użyciem komputera. W takim przypadku korzystne jest zastosowanie rozwiązania, które pozwoli zmierzyć poziom bazowy danej jednostki oraz minimalne i maksymalne wartości mierzonych cech. Może się to odbywać z wykorzystaniem zadania poprzedzającego grę [21] lub podczas poziomu wprowadzającego, gdzie osoba zapoznaje się ze światem gry [22].

W oparciu o uzyskane w ten sposób dane możliwe jest wygenerowanie afektywnych modeli jednostki [60], które charakteryzują aktualny stan danej osoby i jego zmienność. Jest to niezwykle istotne, gdyż w programowaniu afektywnym bardzo ważne są informacje zarówno o nastroju, które mogą wpływać na kontekst, ale też tymczasowe fluktuacje. Te drugie zaś szczególnie w kontekście growych wzorców projektowych [23].

Kolejne podrozdziały zawierają omówienie zagadnienia monitorowania funkcjonowania ludzi, wykorzystując uproszczony podział na dwa ostatnie elementy ze względu na to, że współcześnie metody subiektywne są rzadko stosowane, bądź sprowadzają się do wypełniania przez użytkowników kwestionariusz po badaniu [22]. W publikacji z 2018 roku [63], sposoby mierzenia zostały podzielone na bezpośrednie (obiektywne) i pośrednie (wywodzące się z logowania aktywności w grach).

3.1.1. Metody bezpośrednie

Ekspresja emocji związana jest z aktywnością Autonomicznego Układu Nerwowego, a to z kolei implikuje pojawienie się charakterystycznych zmian w ustroju człowieka [24]. Mogą one być mierzone przy pomocy właściwych sensorów [9, 66]. Odpowiednie bioczujniki umożliwiają także pomiar zachowań związanych z uczuciami.

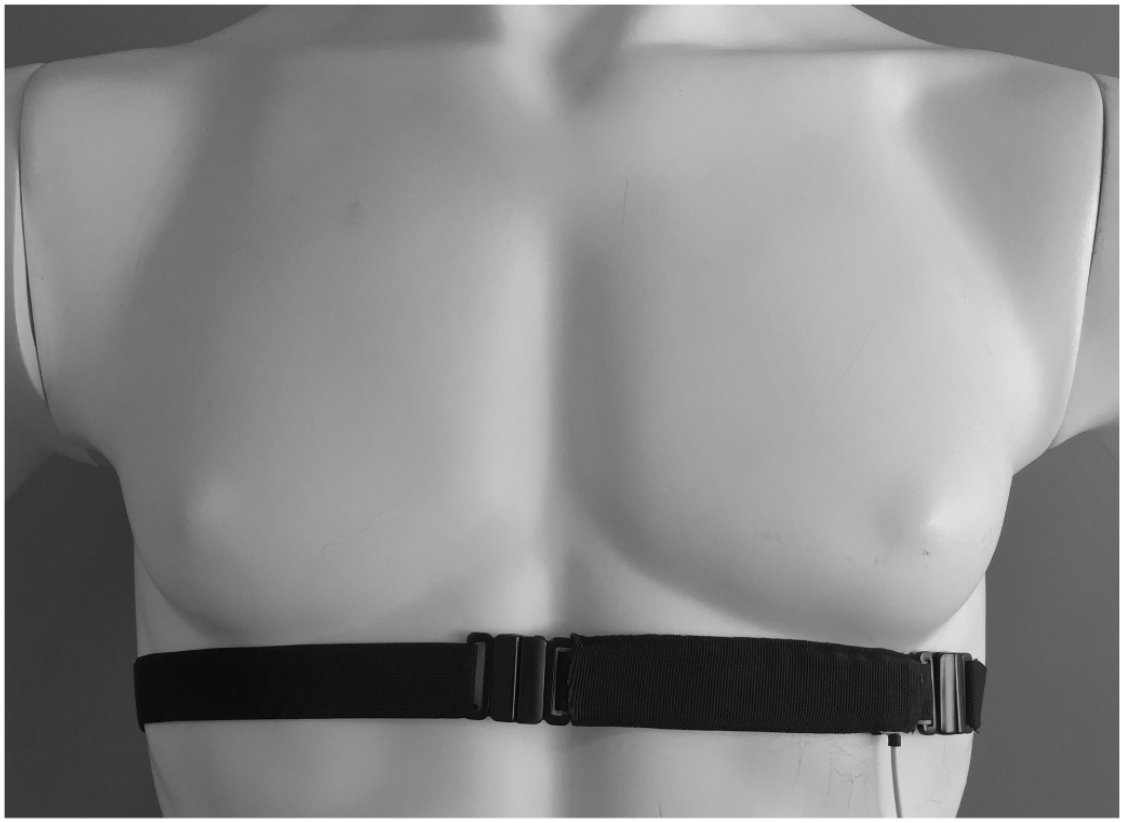
Modalnościami powszechnie wykorzystywanymi w programowaniu afektywnym w ramach paradygmatu obiektywnego są:

1. Tętno oraz ciśnienie krwi [126], zależne od pracy serca.
2. Reakcja elektrodermalna [126], czyli zmiana oporu elektrycznego skóry, zależna od pracy gruczołów potowych i nawilżenia skóry.
3. Respiracja [126], czyli oddychanie, zależne od pracy płuc.
4. Aktywność mięśni [127], zależna od ich skurczów i rozkurczów.
5. Aktywność elektryczna mózgu [9], zależna od jego pracy.
6. Temperatura ciała [128], zależna od procesów wewnętrznych.
7. Ruchy gałek ocznych [129], zależne od kierunku, w który osoba spogląda.

Do zalet metod bezpośrednich należy zaliczyć to, że emocje są mierzone w sposób ilościowy, co z programistycznego punktu widzenia jest bardzo dogodne. Pozwala na opracowanie prostych algorytmów reagujących na progi. Wysoka skuteczność, która wiąże się z tym podejściem jest okupiona częściową inwazyjnością [22]. Jest to szczególnie widocznie w przypadku

rozwiązań laboratoryjnych i znacznie mniej odczuwalne, gdy sprowadza się do tak zwanych urządzeń nasobnych [21].

Opisane w tym akapicie metody, pozwalają także na użycie tak zwanych mechanik fizjologicznych [22]. Są to między innymi sterowanie poprzez mrużenie oczu, czy intensywny wdech i zatrzymywanie powietrza w płucach w celu obrony przed wrogami (rys. 3.2). W przeciwieństwie do mechanik pośrednich, na których bazuje większość aplikacji z zakresu gier afektywnych, wymagają one świadomego użycia przez graczy oraz specjalnego sprzętu umożliwiającego ich rejestrację. U Caminhy zostały ocenione jako najbarwniejsze, najchętniej używane i najbardziej zapadające w pamięć [22].



Rysunek 3.2. *Opaska założona na klatkę piersiową umożliwiająca pomiar respiracji*, źródło: [130]

Każda z wymienionych wyżej modalności doczekała się wielu opracowań opracowań w zakresie programowania afektywnego [79, 63]. Jednak szczególnym zainteresowaniem, niezmiennie od początków istnienia dziedziny, cieszą się reakcja elektrodermalna [131], metody związane z pomiarem pracy układu krążenia [132] i analiza pracy mięśni. Jest to wynik wysokiej skuteczności tych rozwiązań w przypadku odczytywania emocji i sterowania [63]. Dlatego też zostaną tutaj pokrótce omówione.

3.1.1.1. Reakcja elektrodermana

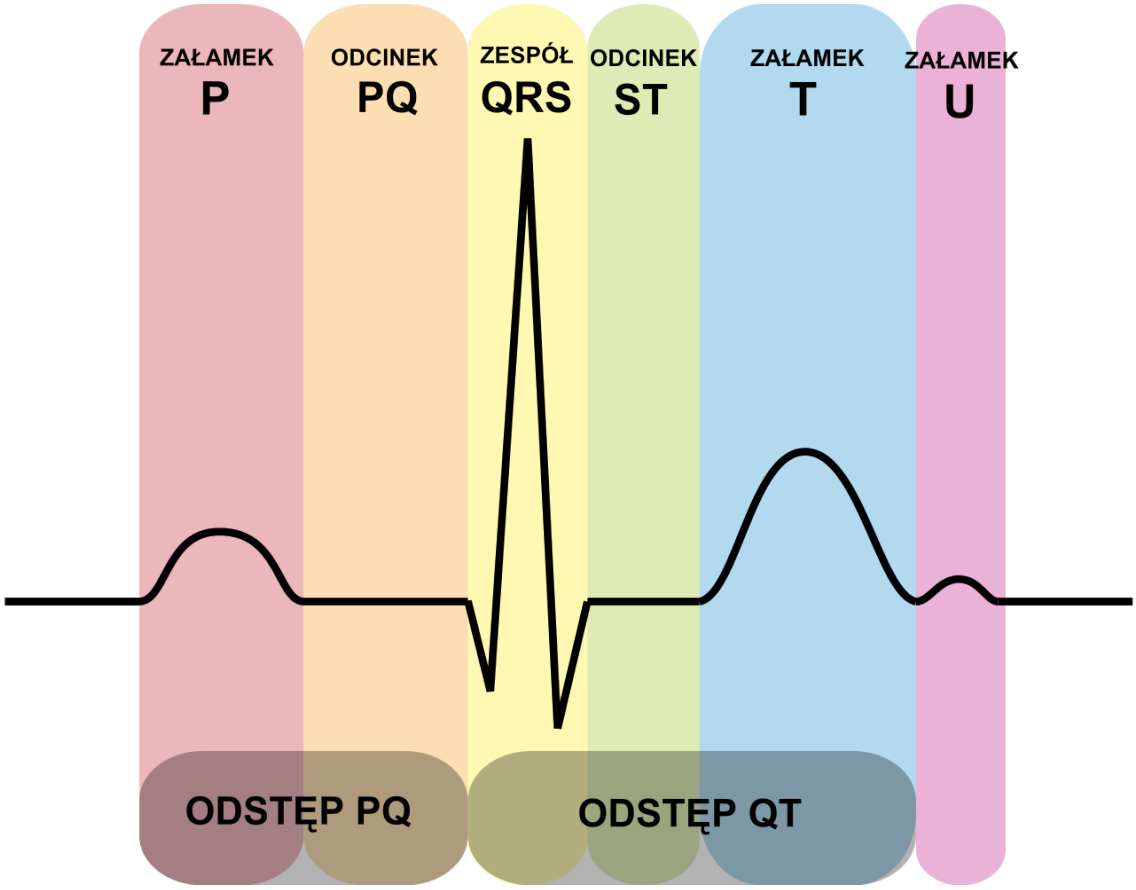
Układ współczulny poprzez pracę gruczołów potowych reguluje stopień nawilżenia skóry [24]. Zjawisko nosi nazwę reakcji skórno-galwanicznej lub reakcji elektrodermalnej. Intensywność wydzielanego potu wpływa w ten sposób na opór elektryczny, który może być mierzony przy pomocy galwanometru (rys. 3.3). Jego dokładność wynosi kilka mikrosimensów, jeśli jest poprawnie skalibrowany. Rejestracja odbywa się z wykorzystaniem dwóch elektrod, które mierzą przepływ prądu elektrycznego generowanego przez ciało bądź przez urządzenie pomiarowe [24, 84].

Rysunek 3.3. *Elektrody przyłożone przytwierdzone do opuszek palców w celu rejestracji reakcji elektrodermalnej*, źródło: [133]

W odniesieniu do badania emocji, przewodnictwo skórne wykorzystuje się koncentrując się na spontanicznej reakcji na niespodziewane bodźce [24]. Ujawnia się zatem najlepiej podczas odczuwania strachu, gniewu, zaskoczenia, zaciekawienia oraz podniecenia seksualnego. Reakcja elektrodermalna jest niezależna od woli, dlatego też bywa wykorzystywana w przypadku wykrywaczy kłamstw.

3.1.1.2. Praca serca

Podstawowym zabiegiem medycznym wykonywanym w celu rozpoznawania chorób serca jest elektrokardiogram. Badanie wykorzystuje elektrody, które są odpowiednio rozmieszczone na tułowiu osoby [24, 84]. Zapisują one depolaryzację i repolaryzację związane z pracą komórek serca. Ich aktywność powoduje pojawienie się charakterystycznych załamków na rejestrowanym wykresie (rys. 3.4). Interpretacja powstałej charakterystyki musi zawsze brać pod uwagę wiele czynników, które nie są związane bezpośrednio z układem krwionośnym, takich jak stosowane leki, czy wiek [24].

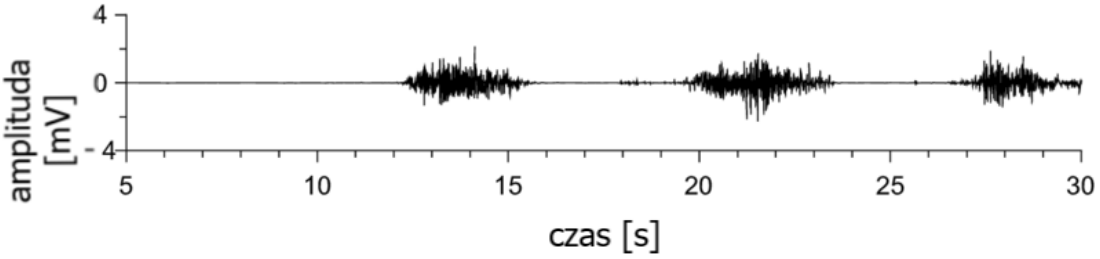


Rysunek 3.4. *Sygnał EKG z odpowiednio zaznaczonymi załamkami, odcinkami i zespołami*, źródło: [134]

Z punktu widzenia programowania afektywnego ważne jest skorelowanie pracy serca z wystąpieniem emocji. Bezpośredni wpływ na aktywność komórek mięśnia sercowego ma układ limbiczny, który odpowiada za regulację zachowań fizjologicznych związanych ze stanami emocjonalnymi i, co więcej, ma wpływ na samo przeżywanie tych stanów [24]. Normowanie związane z uczuciami odbywa się bezpośrednio za pomocą nerwów, które łączą się z sercem i wpływają tym samym na jego aktywność. Jest to szczególnie odczuwalne w przypadku emocji negatywnych oraz intensywnych uniesień.

3.1.1.3. Praca mięśni

Poprzez wzmocnienie potencjałów elektrycznych mięśni za pomocą elektromiogramu możliwa jest diagnoza pracy mięśni oraz nerwów obwodowych. W celu pomiaru wykorzystywane są dwie elektrody umieszczone bezpośrednio na analizowanym mięśniu. Możliwe jest również skorzystanie z trzeciej elektrody jako referencji, którą umieszcza się na widocznej kości łokcia, nadgarstka lub kolana. Dzięki temu w łatwy sposób możliwe jest rozróżnienie (rys. 3.5), kiedy mięsień jest w stanie skurczenia, a kiedy jest rozkurczony.



Rysunek 3.5. *Sygnał EMG, wysoka amplituda oznacza skurcz/rozkurcz, a niska odpowiednio rozkurcz/skurcz*, źródło: [135]

Elektromiografia w przypadku gier afektywnych może być wykorzystana do rozszerzenia istniejących mechanik pozwalając na kontrolę czynności za pomocą sygnałów biologicznych. Można wyobrazić sobie, że zamiast wciskania klawisza myszy, użytkownik korzysta z zgięcia mięśnia palca i ten sposób oddaje strzał. Zastosowanie takich rozwiązań może pozytywnie wpłynąć na poziom immersyjności danej aplikacji.

3.1.2. Metody pośrednie

W związku z ograniczeniami metod bezpośrednich, naukowcy wypracowali szereg sposobów odczytywania stanu emocjonalnego bez konieczności posiłkowania się specjalnymi urządzeniami. Używanie dodatkowych sensorów może wprawiać badanego w dyskomfort, irytację bądź inne stany emocjonalne, dlatego ważne jest wykorzystywanie metod, które ograniczają stosowanie tych praktyk na badanym. Skutkuje to większą naturalnością całego procesu rozrywki afektywnej, jest jednak okupione zmniejszeniem skuteczności i koniecznością opracowywania dokładniejszych modeli emocji.

W ramach paradygmatu pośredniego w programowaniu afektywnym wykorzystuje się:

1. Intensywność kliknięć z wykorzystaniem myszki (rys. 3.6), klawiatury oraz pada [80, 136].

2. Przybieraną posturę [137].

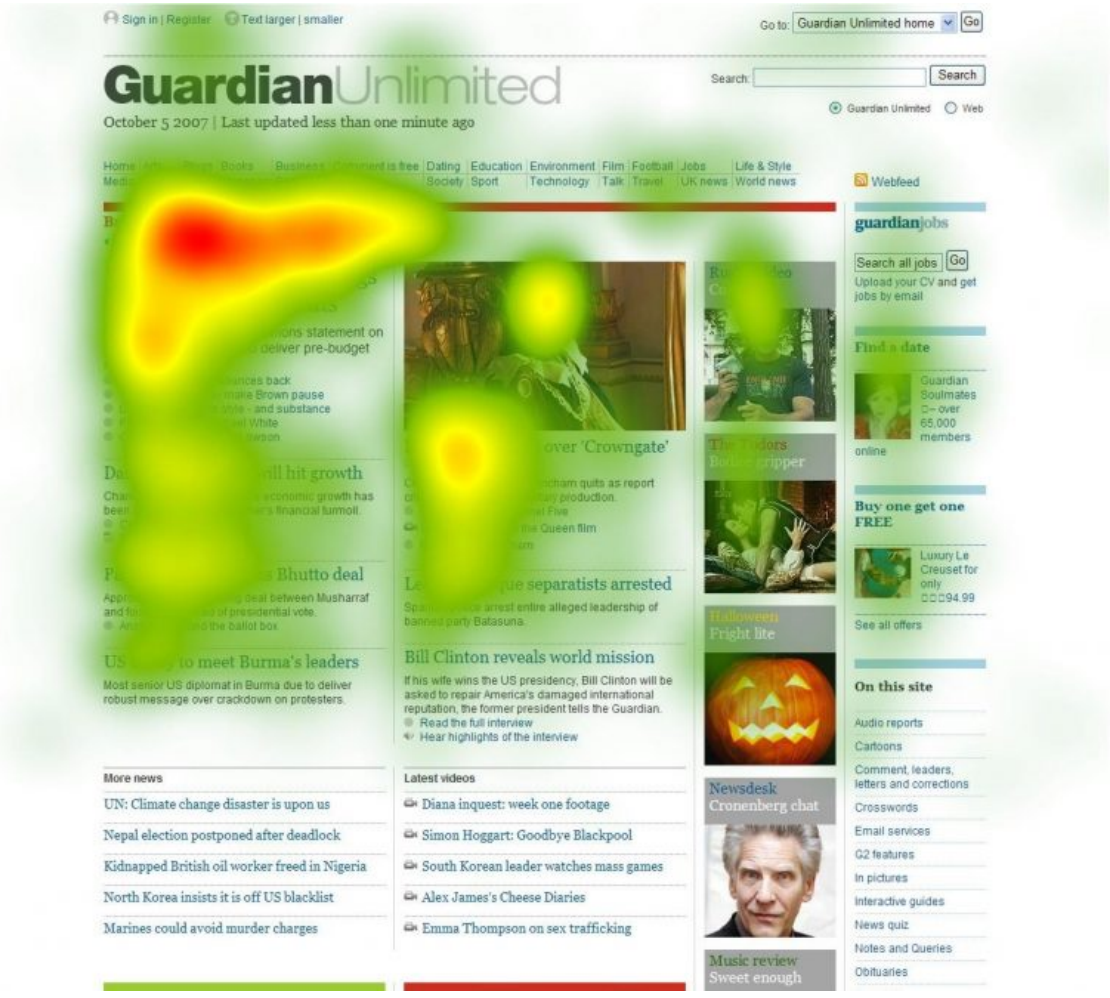
3. Umiejscowienie spojrzenia [138].

4. Cechy głosu [139].

5. Wyraz twarzy [140].

6. Styl gry [141].

Spośród wymienionych wyżej sposobów pomiaru bezpośrednio, pierwsze pięć można zaliczyć do charakterystyk niskopoziomowych, które nie wymagają opracowania złożonych modeli na temat gracza. Podobnie jak metody bezpośrednie koncentrują się na interakcji ze sprzętem [63]. Ostatnia zaś wymaga znacznie bardziej złożonego podejścia. Założenie, że sposób gry zmienia się wraz z emocjami wydaje się być uzasadnione, co widać szczególnie w sytuacji, kiedy gracz musi się skradać, by nie zostać zauważony. Jego styl będzie bardziej wyrafinowany i będzie starać się unikać błędów. Przeciwnie, w sytuacji, kiedy napierają na niego wrogowie, wybierze on bardziej agresywną i bezpośrednią formę rozgrywki.



Rysunek 3.6. *Mapa kliknięć na stronie internetowej*, źródło: [142]

Opracowywanie takich scenariuszy i przewidywanie zdarzeń może się okazać bardzo pomocne z punktu widzenia osoby tworzącej zarysy gier komputerowych. Warto tutaj ponownie zwrócić uwagę na afektywne growe wzorce projektowe [21], które mogą tworzyć podwaliny do przygotowania schematów. Rozpoznając elementy niższego stopnia, można przejść do bardziej złożonego etapu. Co więcej, wykorzystanie wysokopoziomowych wzorców może owocować

zrozumieniem, dlaczego pewne motywy są tak chętnie inkorporowane przez twórców. Mowa tu na przykład o budzącym grozę schodzeniu do piwnicy opuszczonego domu, w którym wszystko skrzypi i trzeszczy lub wywołującym dreszcz skradaniu się, by ukraść potrzebny towar.

3.2. Platformy sprzętowe

Pomiar sygnałów fizjologicznych to jeden z najważniejszych elementów pętli afektywnej. Zebranie danych, a następnie ich dekodowanie i analiza pozwalają wnioskować na temat stanów emocjonalnych, których doświadcza jednostka [9, 66]. Proces akwizycji danych w kontekście badań naukowych z zakresu informatyki afektywnej odbywa się głównie z wykorzystaniem urządzeń klasy medycznej [22]. Warto zauważyć nową tendencję w obrębie tej dziedziny. Uczeni coraz częściej sięgają do urządzeń nasobnych, które często wykazują podobny poziom zaawansowania w zakresie rejestracji sygnałów fizjologicznych jak aparatura profesjonalna, a przy tym charakteryzują się większą mobilnością, umożliwiając swobodę ruchów [84]. Wykorzystanie takich urządzeń może skutkować popularyzacją trendów z dziedziny programowania afektywnego wśród ogółu społeczeństwa.

Rysunek 3.7. *Urządzenie klasy medycznej NeXus-10*, źródło: [143]

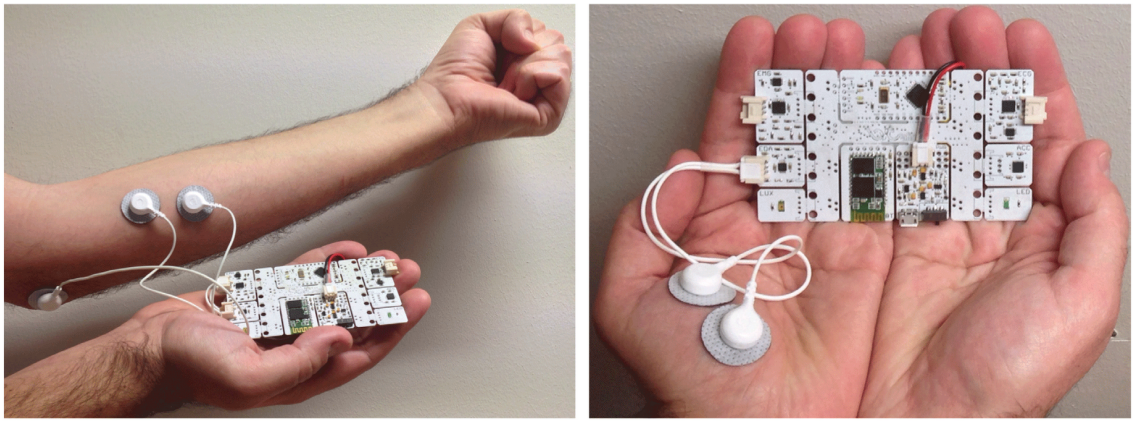
Do platform klasy medycznej można zaliczyć miedzy innymi urządzenia Neurobit Optima oraz NeXus-10 (rys. 3.7). Oba charakteryzują się możliwościami pomiaru pracy serca, mózgu reakcji elektrodermalnej, temperatury i wielu innych [144, 143]. Ogromnym plusem tych zastosowań ze względu na przenośność jest wykorzystanie technologii Bluetooth. Nie zmienia to jednak faktu, że samo urządzenie cechuje się sporymi rozmiarami, a co gorsze jego waga jest znaczna. W obu przypadkach można korzystać z oprogramowania dostępnego u producenta

oraz z interfejsu programistycznego, który umożliwia tworzenie własnych rozwiązań [144, 143]. Do zalet należy wysoka skuteczność, stabilność i rzetelność pomiarowa potwierdzona certyfikatami medycznymi.

Rysunek 3.8. *Apple Watch umożliwia nie tylko podgląd godziny i powiadomień, ale także gromadzenie informacji o aktywności użytkownika*, źródło: [84]

Najwygodniejszym rozwiązaniem z punktu widzenia programowania afektywnego wydają się inteligentne opaski oraz zegarki [84]. Dużą popularnością cieszą się Xiaomi Mi Band, Apple Watch (rys. 3.8) oraz Microsoft Band [145]. Większość z nich potrafi rejestrować puls oraz reakcję elektrodermalną, czasami posiadają wbudowane sensory ruchu, które umożliwiają zliczanie kroków. Wyjątkiem jest tutaj Empatica E4 opracowana przez firmę Picard, która potrafi rejestrować temperaturę [84]. Ogromnym minusem przedstawionych rozwiązań jest niedokładność pomiarów w porównaniu ze sprzętem klasy medycznej. Jak wykazano, część sygnałów jest albo nierejestrowana, albo rejestrowana błędnie [145]. Co więcej, producenci tych urządzeń zazwyczaj udostępniają jedynie dedykowane aplikacje umożliwiające akwizycję danych, a tworzenie własnych rozwiązań jest utrudnione przez ubogą dokumentację bądź brak interfejsu programistycznego [84].

Rozwiązanie pośrednie pomiędzy omówionymi wyżej klasami przynoszą platformy takie jak e-Health Sensor Platform [146] oraz BITalino (r)evolution kit [147]. Pierwsze z nich to rozszerzenie popularnego układu Arduino Uno. Dzięki bibliotece dostarczanej przez producenta komunikacja z urządzeniem odbywa się za pomocą interfejsu napisanego w języku C [146]. Drugie (rys. 3.9) z nich to samodzielna platforma [147] z osobnymi narzędziami programistycznymi dostępnymi na ogromną ilość popularnych rozwiązań, takich jak najpopularniejsze systemy operacyjne typu Android i iOS, konkretne implementacje w wysokopoziomowych językach jak Python i C#, a nawet konkretne środowiska, w tym silniki gier jak Unity.



Rysunek 3.9. *Platforma BITalino z sensorami do pomiaru aktywności mięśni i reakcji elektrodermalnej*, źródło: [147]

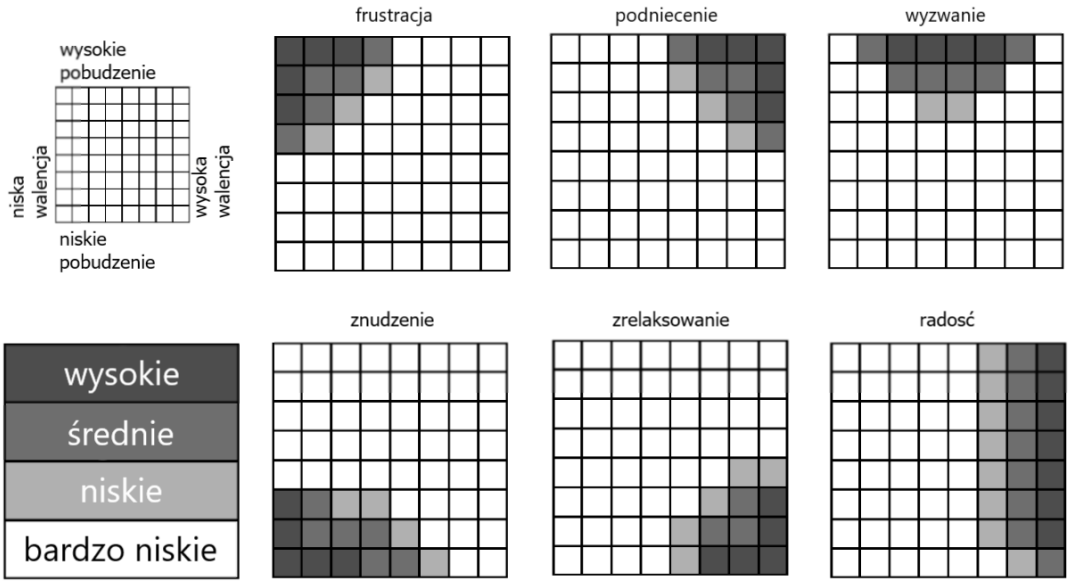
Obie platformy umożliwiają pomiar pracy serca, reakcji elektrodermalnej, temperatury, stężenia glukozy we krwi oraz wielu innych sygnałów fizjologicznych [146, 147]. Wykazują się skutecznością zbliżoną do rozwiązań laboratoryjnych, cechując się przy tym znacznie większą lekkością i mobilnością [84]. Główną przewagą platformy BITalino (r)evolution kit jest wbudowana komunikacja [147] za pomocą Bluetooth, która jest domyślnie niedostępna w przypadku e-Health Sensor Platform.

3.3. Techniki rozpoznawania emocji

Korzystając z danych na temat fizjologii i zachowania graczy opracowano wiele metod, które umożliwiają rozróżnianie uczuć. Bardzo pomocne okazują się tutaj sposoby korzystające ze sztucznej inteligencji. Jednym z najważniejszych etapów na drodze do uzyskania systemu zdolnego do rozpoznawania emocji jest wytrenowanie go w oparciu o wcześniej przygotowany zbiór uczący. Zebranie danych jest karkołomnym przedsięwzięciem i często zajmuje najwięcej czasu projektantom układów sztucznej inteligencji. Fundamentem tego procesu jest taki wybór,

a następnie ekstrakcja cech danych, żeby system w oparciu o wskazane właściwości był w stanie wykonać pożądaną czynność.

Wyodrębnienie odpowiednich charakterystyk jest zazwyczaj podparte metodami statystycznymi albo wykorzystywaniem modeli, które obrazują dane zjawisko. W wypadku stanów emocjonalnych i programowania afektywnego jedną z częściej stosowanych teorii [21, 22, 63], jest model emocji stworzy przez Russella [64]. Operuje on na dwóch charakterystykach: walencji, która jest miarą przyjemności danego stanu oraz pobudzeniu, które określa intensywność przeżywanego uczucia. Badania wskazują, że schemat Russella jest częściowo pozytywnie skorelowany z poszczególnymi emocjami podstawowymi i może być pomocny przy ich rozróżnianiu. Warto wspomnieć tutaj o dwuwymiarowej przestrzeni [22, 64], w której każdej emocji przypisano odpowiednie wartości walencji i pobudzenia. W ten sposób frustracja jest związana z niewielką wartością pierwszej z nich i intensywną stymulacją [22].



Rysunek 3.10. *Wykorzystanie dwuwymiarowej przestrzeni walencji i pobudzenia do wnioskowania na temat stanów emocjonalnych*, źródło: [22]

Bazując na dwuwymiarowym modelu Russella [64], Mandryk i Atkins przeprowadzili serię eksperymentów [148] w celu rozczytania emocji graczy produkcji NHL 2003. Rejestrowano reakcję elektrodermalną, pracę serca i mózgu, a następnie wykorzystano logikę rozmytą, by wyodrębnić poszczególne emocje (rys. 3.10). Caminha, bazując na tej pracy stworzył system implementujący pętlę afektywną [22]. Udało się znaleźć pewne korelacje, jednak, jak wskazują autorzy, użycie logiki rozmytej mogło spowodować, że pewne subtelności zostały pominięte.

Logika rozmyta opiera się na reprezentacji zmiennych pewnymi określeniami, a nie dokładnymi liczbami. Pomysł czerpie garściami z ludzkiej mowy potocznej. Precyzyjne wartości przy pomocy tak zwanych funkcji przynależności zostają zamienione na określenia lingwistyczne, takie jak „niski", „średni" oraz „wysoki". Jak w tradycyjnej logice, tworzone zostają reguły [148] albo drzewa decyzyjne [149], na podstawie których można prowadzić wnioskowanie. Szczególnie nadają się więc do analizy sygnałów ciągłych, które są potencjalnie zaszumione [148]. Z tego względu logika rozmyta jest często uznawana za naturalnego kandydata do analizy sygnałów fizjologicznych [22].

Do bardziej wyrafinowanych metod sztucznej inteligencji zaliczamy między innymi sieci neuronowe o głębokiej strukturze [150], algorytmy genetyczne [151], drzewa decyzyjne [152], czy maszyny wektorów nośnych [153]. Bardzo często działają one w oparciu o skomplikowane przekształcenia statystyczne. Przykładem może być Google Jigsaw [154], które dzięki analizie komentarzy zostawionych przez użytkowników na forach pod artykułami Wikipedii umożliwia rozpoznawanie kontekstu semantycznego wypowiedzi i przeciwdziałanie pojawieniu się *mowy nienawiści* na łamach portalu.

4. Zakres pracy

4.1. Cele pracy

Celem pracy jest opracowanie metod i narzędzi inżynierii oprogramowania wspierających budowanie gier z użyciem growych wzorców projektowych na platformie Unity.

4.2. Założenia projektu

1. Przegląd aktualnej literatury oraz rozwiązań związanych z tematyką:
   (a) Interakcji między człowiekiem a komputerem.
   (b) Projektowania gier komputerowych z uwzględnieniem metod programowania afektywnego.
   (c) Sposobów monitorowania i analizy zachowań graczy.
2. Selekcja growych wzorców projektowych, które mogą mieć znaczenie afektywne.
3. Wybór mechanik umożliwiających stworzenie afektywnej gry komputerowej.
4. Opracowanie projektu afektywnej gry komputerowej, w oparciu o zestaw wzorców i mechanik.
5. Implementacja stworzonego projektu afektywnej gry komputerowej w środowisku Unity.
6. Wykonanie aplikacji umożliwiającej akwizycję sygnałów fizjologicznych przy użyciu platformy BITalino (r)evolution kit.
7. Domknięcie pętli afektywnej.
8. Ewaluacja stworzonego rozwiązania.

4.3. Poruszone zagadnienia

Zagadnienia z zakresu interakcji pomiędzy człowiekiem a komputerem, programowania afektywnego, tworzenia gier elektronicznych oraz technik monitorowania i wnioskowania o stanie emocjonalnym człowieka tworzą podwaliny niniejszej pracy. Zebranie i opracowanie wątków teoretycznych pozwoliło na jednoznacznie sprecyzowanie problemów, które będą poruszone w części praktycznej tego projektu. Zostały one opracowane powyżej jako cele i założenia towarzyszące autorowi.

Analizując dokonania badaczy z zakresu tworzenia gier elektronicznych, ustalono, że w pracy opisany będzie proces budowania afektywnej gry komputerowej. Ważnym elementem pracy będzie zaprezentowanie podejścia projektowego, które uwzględnia wykorzystanie growych wzorców projektowych jako metod i narzędzi charakterystycznych dla rozwiązań inżynierii oprogramowania.

Rysunek 4.1. *Zdziwienie na twarzach starszej pary grającej wspólnie w grę komputerową*, źródło: [155]

Temat programowania afektywnego poruszono ze względu na silny rozwój metod interakcji pomiędzy człowiekiem a komputerem. Podejście oparte na emocjach obiecuje tworzenie bardziej naturalnych interfejsów bez konieczności inwestowania w drogie platformy takie jak okulary wirtualnej rzeczywistości, czy projektory. Tym bardziej, że to właśnie emocje stanowią najbardziej naturalną formę komunikacji wśród ludzi [39, 45]. Podstawą do wyboru tematyki gier komputerowych jest to, że najpełniej spełniają one założenia informatyki afektywnej ze względu na możliwość łatwego wzbudzania (rys. 4.1) stanów emocjonalnych [21].

Aby uwypuklić planowanie gier, badania rozpoczną się od stworzenia puli afektywnych wzorców projektowych ze zbioru przedstawionego przez Björka i Holopainena [23] oraz późniejszych twórców [20, 21, 22]. Pozwoli to na sporządzenie struktury gry komputerowej oraz zrozumienie, które elementy gry mogą budzić zmiany afektywne.

Posługując się opracowaniem zaprezentowanym przez Gilleade, Dix oraz Allanson [97] uwaga zostanie zwrócona szczególnie na te mechaniki, które umożliwiają stworzenie gry trudniejszą i bardziej wyzywającą dla gracza. Korzystanie z tych wzorców będzie oparte na kontinuum relaksu i znudzenia z jednej strony oraz stresu i frustracji z drugiej.

Wnioskowanie na temat emocji zostanie przeprowadzone przy pomocy logiki rozmytej [22] oraz modelu opartego o przestrzeń walencji i pobudzenia [64]. Jest to podyktowane prostotą wskazanej metody oraz tym, że rozwiązania te z dużą skutecznością odwzorowują emocje. Niejednokrotnie wskazywano na korelację sygnałów fizjologicznych związanych z pracą serca oraz reakcji elektrodermalnej z wymienionymi powyżej uczuciami [156]. Wskazano także, że zmiany afektywne mogą pojawiać się jako następstwo growych wzorców projektowych [123]. Dlatego też interesujące wydaje się rozpoczęcie procesu projektowania od wybrania mechanik związanych ze zmianami fizjologicznymi w ciele ludzi.

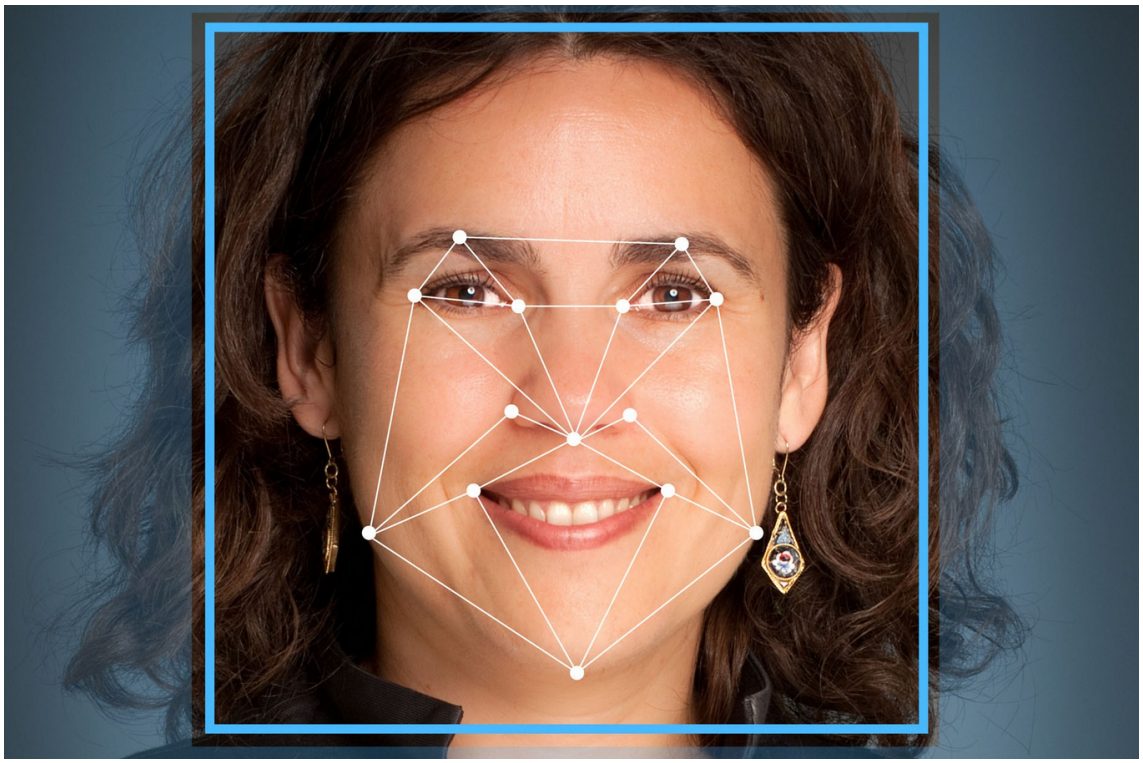
W celu osiągnięcia jak największej skuteczności rozpoznawanie emocji będzie bazować na metodach bezpośrednich [63], a zatem opartych o wykorzystanie odpowiedniego urządzenia (rys. 4.2). Sygnały pochodzące z ciała będą mierzone przy pomocy platformy sprzętowej BITalino (r)evolution kit [147].

Decyzja o wybraniu tego przyrządu podyktowana jest niewielkimi rozmiarami. Będzie miało to wpływ na niską inwazyjność rozwiązania, a co za tym idzie brak przekłamań w kontekście stanów emocjonalnych. Istotna jest także możliwość akwizycji dokładnych i powtarzalnych danych pomiarowych [84]. Ponadto, dzięki komunikacji Bluetooth oraz szerokiemu wsparciu dla różnych rozwiązań programistycznych, platforma BITalino (r)evolution kit stanowi doskonały wybór w zakresie programowania afektywnego.

Jednym z elementów systemu będzie opracowanie skutecznej mechaniki, która pozwoli wykorzystać bezpośrednie sprzężenie afektywne przy pomocy elektromiografii.

Równolegle, bazując na wcześniej opracowanym projekcie zostanie zaprogramowana gra komputerowa. Kluczową rolę odegra tutaj silnik Unity. Wybór tego narzędzia podyktowany jest przede wszystkim ogromną liczbą materiałów edukacyjnych, wsparciem społeczności i dostępnością darmowych rozwiązań za pośrednictwem sklepu internetowego. Warto wskazać też możliwości generacji wielu wersji tej samej gry tak, by mogły one działać na różnych systemach. Nie bez znaczenia pozostaje również integracja Unity z platformą BITalino [147, 114], co pozwoli na skuteczne domknięcie pętli afektywnej.

W efekcie wykrywanie stanów znudzenia lub zrelaksowania będzie skutkować aktywacją afektywnych growych wzorców projektowych. Mają one utrudniać graczowi rozgrywkę, co może być obserwowalne przez pojawienie się frustracji bądź stresu. Szczególnym elementem będzie pokazanie metod inżynierii oprogramowania takich jak growe wzorce projektowe w kontekście budowania aplikacji.



Rysunek 4.2. *Jedna z metod bezpośrednich, wnioskowanie na temat emocji za pomocą punktów charakterystycznych twarzy przy pomocy kamery*, źródło: [157]

Bazowanie na nich oraz wykorzystanie urządzeń, które nie są klasy medycznej w celu generacji pętli afektywnej jest podejściem, którego nie można spotkać w obecnej literaturze informatycznej. Podobnie jak w badaniach Caminhy [22], za sukces będzie można uznać stworzenie takiej gry, która zostanie dobrze odebrana przez badanych.

5. Freud Me Out

5.1. Podstawowa struktura

W ramach niniejszej pracy magisterskiej stworzono grę elektroniczną, która realizuje założenia programowania afektywnego poprzez implementację pętli sprzężenia zwrotnego. Całość aplikacji wykorzystuje wymagania i analizę tych kryteriów, które szczegółowo omówiono w poprzednim rozdziale.

Podstawową strukturę gry komputerowej oparto o wcześniej wyselekcjonowane wzorce projektowe na podstawie opracowania Björka i Holopainena [23]. Tworzą one fundament aplikacji i dlatego w tej sekcji wyszczególniono każdy z nich i omówiono jego wykorzystanie.

Program przeznaczono tylko dla jednej osoby. Wykorzystano zatem wzorzec **gry jednoosobowej**. Umożliwia to lepsze kontrolowanie warunków eksperymentów przeprowadzonych później w celu ewaluacji rozwiązania.

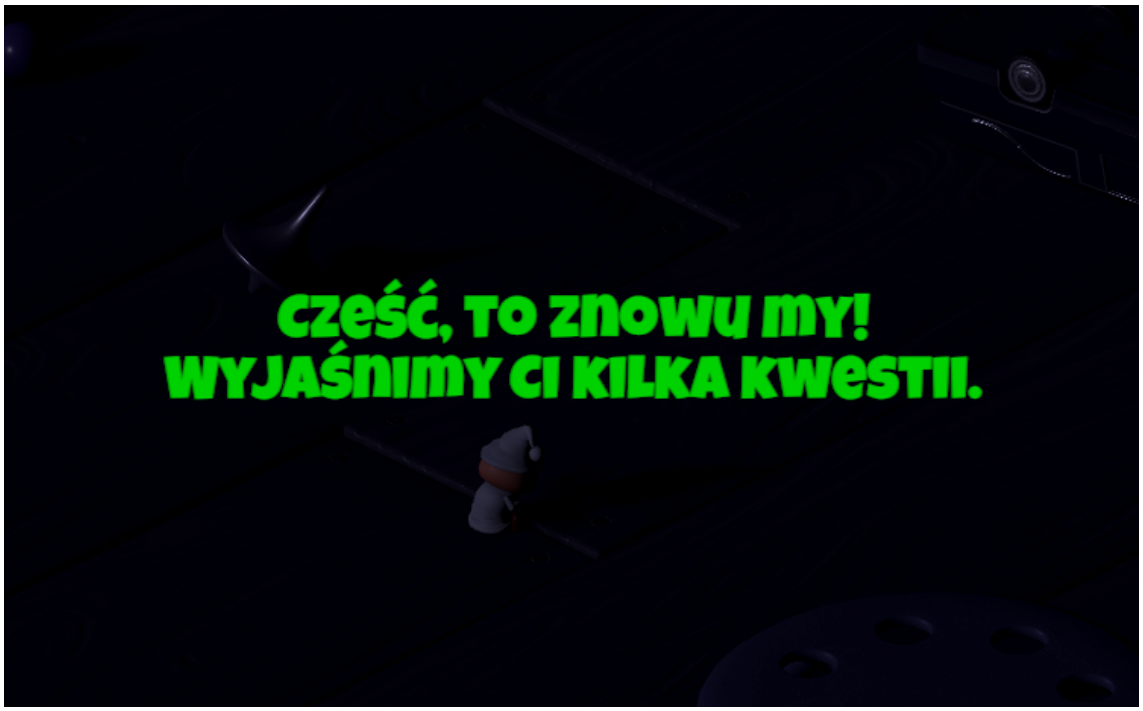
Projekt składa się z pięciu zróżnicowanych pod względem trudności **poziomów**. Każdy z nich poprzedzony jest krótki opowieściami. Po zebraniu stanowią one spójną **historię**. Specyfika **narracji** gry skupia się wokół środowiska psychologicznego. Zasiadając przed komputerem gracz wciela się w rolę osoby, która ma problemy z pamięcią. Aby się ich pozbyć, udaje się na niekonwencjonalną terapię, która czerpie z ujęcia psychoanalitycznego i prac Zygmunta Freuda. Leczenie polega na dosłownym przeniesieniu się do **alternatywnej rzeczywistości**, czyli do środka psychiki. Wykorzystano odwołanie się do wnętrza osoby. Stąd również wzięła się nazwa aplikacji, czyli *Freud Me Out*.

Rysunek 5.1. *Fragment interfejsu użytkownika, po lewej poziom życia a po prawej poziom energii,* źródło: opracowanie własne.

Pierwszy poziom stanowi wprowadzenie do **świata gry**. Umożliwia to zapoznanie się z podstawowymi konceptami **poruszania się** i **walki**. Omówione zostały też elementy **interfejsu**

użytkownika takie jak poziom energii, poziom zdrowia (rys. 5.1) oraz ilość punktów. Pierwsze dwa elementy przedstawiono przy pomocy pasków, ostatni za pomocą komunikatu słownego. Staje się on coraz bardziej entuzjastyczny w miarę uzyskiwania lepszego rezultatu. Obrazuje to wykorzystanie wzorca **niedokładnej informacji**.

Następne fazy gry to kolejne, coraz głębsze elementy psychiki. Wyszczególniono świadomość, przedświadomość oraz nieświadomość. Na każdym z tych poziomów gracz walczy z **przeciwnikami** symbolizującymi konkretne wspomnienia. Króliki reprezentują dokarmianie zwierząt w dzieciństwie, pluszowe misie pierwszego wymyślonego przyjaciela, a słonie stanowią istotny fragment wydarzenia, z którego główny bohater ledwo uszedł z życiem. Ostatni etap to walka z głównym lękiem będącym źródłem problemów z pamięcią. Przybiera on postać **bossa**, czyli najważniejszego antagonisty.

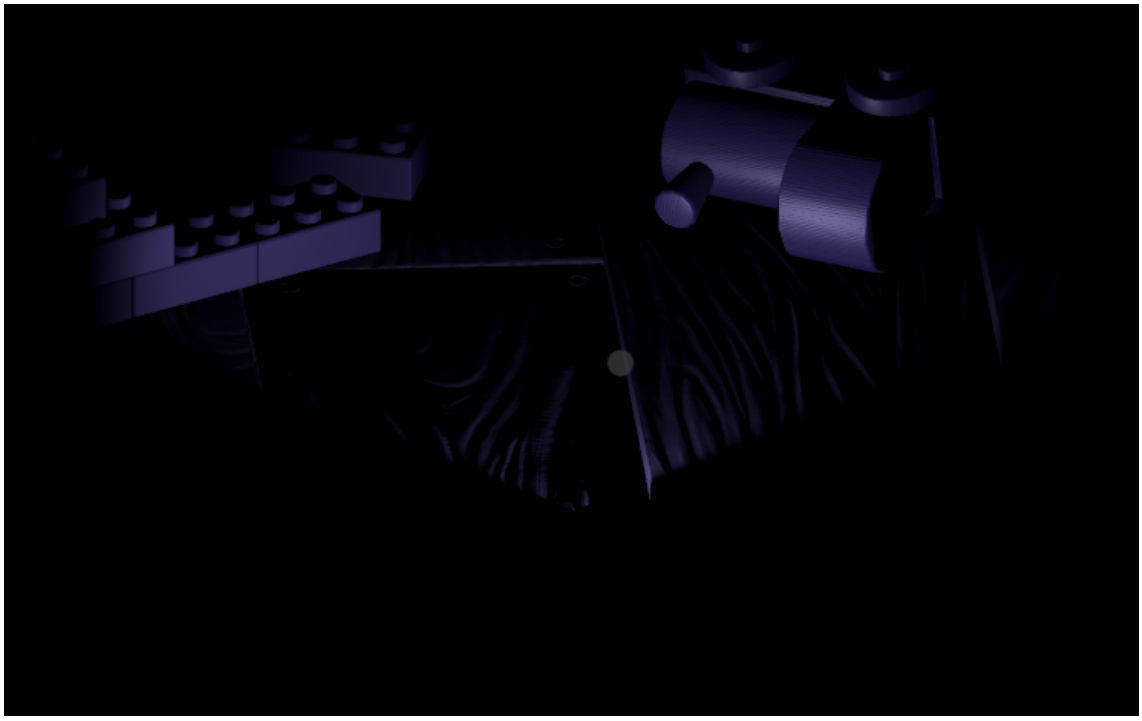


Rysunek 5.2. *Komunikaty przekazywane przez terapeutów,*
źródło: opracowanie własne.

Przez całą rozgrywkę bohaterowi towarzyszą terapeuci, którzy wykazując się poczuciem humoru i ironii komentują poczynania swojego pacjenta, ostrzegają go o niebezpieczeństwach oraz opowiadają o sposobach interakcji z aplikacją. Dzięki temu każda czynność podjęta przez gracza niesie za sobą **przewidywalne konsekwencje**. Wypowiedzi wyświetlane są przy pomocy **alarmów**, czyli tekstów znajdujących się na środku ekranu (rys. 5.2) i przysłaniających część rozgrywki.

Odwołanie się do psychologii ma podwójny charakter. Po pierwsze, gra mimo tego, że jest okraszona zabawnymi komentarzami, opowiada o prawdziwych teoriach. Wskazuje to **edukacyjny** wymiar rozwiązania. Osoby, które nie są zaznajomione z pracami Freuda mogą znaleźć w niej podstawowe informacje na temat psychoanalizy i struktury samej świadomości. Po drugie, nawiązanie do psychologii jest też widoczne w sposobie oddziaływania na grę ze względu na pętlę afektywną.

Na wszystkich pięciu poziomach gracz, którego **awatarem** jest dziecko ubrane w koszulę nocną, znajduje się w tym samym pomieszczeniu, czyli pokoju z zabawkami. Świat gry stanowi zatem odwołanie do dzieciństwa bohatera, które we wspomnieniach wielu ludzi ogranicza się do tego niewielkiego pomieszczenia. Na podłodze rozmieszczone są zabawki, które należy **omijać**. Stylistykę gry osadzono w mrocznej scenerii horroru. Mowa tu również o budzącej dreszcz muzyce.



Rysunek 5.3. *Widok pierwszoosobowy na trzecim poziomie gry, interfejs użytkownika został dezaktywowany,* źródło: opracowanie własne.

Przestrzeń jest oglądana z **widoku izometrycznego**. Jego najistotniejszą cechą jest to, że między wszystkimi parami osi znajdują się kąty o mierze 120 stopni. Poziom trzeci przynosi zmianę widoku, co jest uzasadnione problemami technicznymi z komunikacją z terapeutami poprzez przeniesienie się do nieświadomości. Gracz ogląda rozgrywkę z **widoku pierwszoosobowego**, widząc jedynie końcówkę broni podczas oddawania strzału i celownik (rys. 5.3). Wyłączone zostają także wszystkie elementy interfejsu.

Co jakiś czas w **losowym** miejscu pokoju **pojawiają się** nowi przeciwnicy, których należy wyeliminować za pomocą **strzału** z pistoletu bądź dodatkowych umiejętności.

Z każdym kolejnym poziomem antagoniści są coraz silniejsi, gdyż po zderzeniu z bohaterem odbierają mu coraz więcej punktów życia. Wraz z kolejnymi fazami rozgrywki także bohater podlega **rozwojowi**, zyskując kolejne punkty życia oraz **umiejętność specjalną**, którą nazwano *SuperMoc*. Dzięki tej zdolności bohater w jednym momencie jest w stanie zneutralizować dużą liczbę przeciwników naraz. Aby zrównoważyć poziom gry, użycie *SuperMocy* jest możliwe jedynie wtedy, kiedy bohater ma wystarczającą ilość energii. Co więcej, wykorzystanie tej zdolności powoduje, że przez określony czas strzały oddawane przez bohatera mają mniejszy zasięg, są słabsze i rzadsze.

Każdy unicestwiony przeciwnik ma przypisaną wartość punktową, która dodawana jest do ogólnego **wyniku** gracza. Po zakończonej rozgrywce możliwe jest porównanie swojego rezultatu z osiągnięciami innych (rys. 5.4), gdyż wyświetlana jest **tablica wyników**.

Rysunek 5.4. *Punkty zdobyte przez najlepszych graczy, wyświetlone po zakończeniu gry,* źródło: opracowanie własne.

W celu osiągnięcia zwycięstwa gracz może stosować jedynie wymienione wyżej akcje. Ich zbiór jest **ograniczony**, nie przewidziano możliwości wykonywania operacji według własnego pomysłu.

5.2. Wymiar afektywny

Wśród wzorców projektowych wyszczególnionych w powyższej sekcji znajduje się bardzo wiele elementów, które mogą mieć znaczenie w kwestii wywoływania stanów afektywnych. Wyróżniono następujące schematy:

1. Historia.
2. Narracja.
3. Alternatywna rzeczywistość.
4. Niedoskonała informacja.
5. Poruszanie się.
6. Omijanie.
7. Przeciwnicy.
8. Walka.
9. Boss.
10. Losowość.
11. Alarmy.
12. Rozwój bohatera.
13. Widok pierwszoosobowy.
14. Tablica wyników.

Wykorzystanie wymienionych niskopoziomowych wzorców pozwala na wytworzenie się schematów z wyższych poziomów. Wygenerowane zostały:

1. **Immersja**, czyli intensywne odczuwanie gry przez użytkownika. Jest przede wszystkim konsekwencją współwystępowania wzorca historii, narracji oraz alternatywnej rzeczywistości. Jednak również pozostałe schematy mogą mieć na nią wpływ, gdyż jest kwestią indywidualną danej osoby.

2. **Odczucie wpływu na świat gry**, czyli wrażenie, że czyny gracza mają konsekwencje w rzeczywistości gry. Występuje jako następstwo współistnienia wzorca walki, przeciwników, bossa oraz rozwoju bohatera.

3. **Identyfikacja**, czyli utożsamianie się gracza z awatarem w świecie gry. Może być szczególnie widoczne w kontekście współczucia względem wspomnianych wyżej problemów z pamięcią. Pojawia się jako konsekwencja jednoczesnego występowania wzorca historii, narracji oraz widoku pierwszoosobowego.

Oprócz wskazanych elementów, zaproponowano również własne rozwiązania, które działając w oparciu o dane fizjologiczne pozwalają na wpływanie na świat gry. Wskazano na interakcję bezpośrednią i pośrednią. Dla celów ewaluacyjnych przygotowano też równoważne mechaniki, które odpowiedniki działające bez używania sygnałów pochodzących z ciała człowieka. Wyróżniono następujące elementy:

1. **Dostosowanie szybkości poruszania się przeciwników pod wpływem poziomu uderzeń serca na minutę**. Pozwala to na utrudnienie rozgrywki graczom znudzonym oraz jej ułatwienie użytkownikom zestresowanym. Skorzystano z korelacji wzrostu pobudzenia ze wzrostem tętna człowieka. Mechanika umożliwia też kontrolowanie poziomu szybkości przeciwników przez próbę bezpośredniego wpływu na charakterystykę pracy własnego serca. Może być to obserwowane podczas zwiększenia lub zmniejszenia częstotliwości oddechów. Odpowiednikiem wzorca w wersji bez pętli afektywnej jest **stałe zwiększanie się szybkości poruszania się przeciwników wraz z upływającym czasem**.

2. **Aktywacja dodatkowych miejsc pojawiania się przeciwników pod wpływem poziomu uderzeń serca na minutę oraz poziomu reakcji elektrodermalnej**. Wykorzystano tu korelację między tymi sygnałami fizjologicznymi a przestrzenią pobudzenia. W przypadku, gdy osoba jest zbyt zestresowana, mechanika nie jest aktywowana. Odpowiednikiem wzorca w wersji bez pętli afektywnej jest **zależna od czasu aktywacja dodatkowych miejsc pojawiania się przeciwników**.

3. **Dostosowanie losowości miejsc pojawiania się przeciwników pod wpływem poziomu uderzeń serca na minutę oraz poziomu reakcji elektrodermalnej**. Ponownie wykorzystano korelację między wskazanymi sygnałami a kontinuum pobudzenia. Im większy stres lub frustracja, tym większe ograniczenie losowości pojawiania się przeciwników. Odpowiednikiem wzorca w wersji bez pętli afektywnej jest **całkowita losowość miejsc pojawiania się przeciwników**.

4. **Użycie *SuperMocy* poprzez napięcie bicepsa**, czyli aktywacja zdolności specjalnej pod wpływem skurczu mięśnia dwugłowego ramienia. Co więcej, im dłużej trwa zgięcie ręki, tym intensywniejszy jest rozbłysk światła, które sygnalizuje aktywację zdolności. Odpowiednikiem wzorca w wersji bez pętli afektywnej jest **użycie *SuperMocy* przy pomocy naciśnięcia odpowiedniego klawisza**.

Wszystkie cztery wzorce w wersji afektywnej, które wymieniono powyżej mogą skutkować zwiększeniem intensywności występowania wzorca immersji, odczucia wpływu na świat gry oraz identyfikacji z głównym bohaterem. Jest to wynikiem tego, że mechaniki do działania potrzebują sygnałów fizjologicznych, a te pochodzą bezpośrednio z ciała gracza.

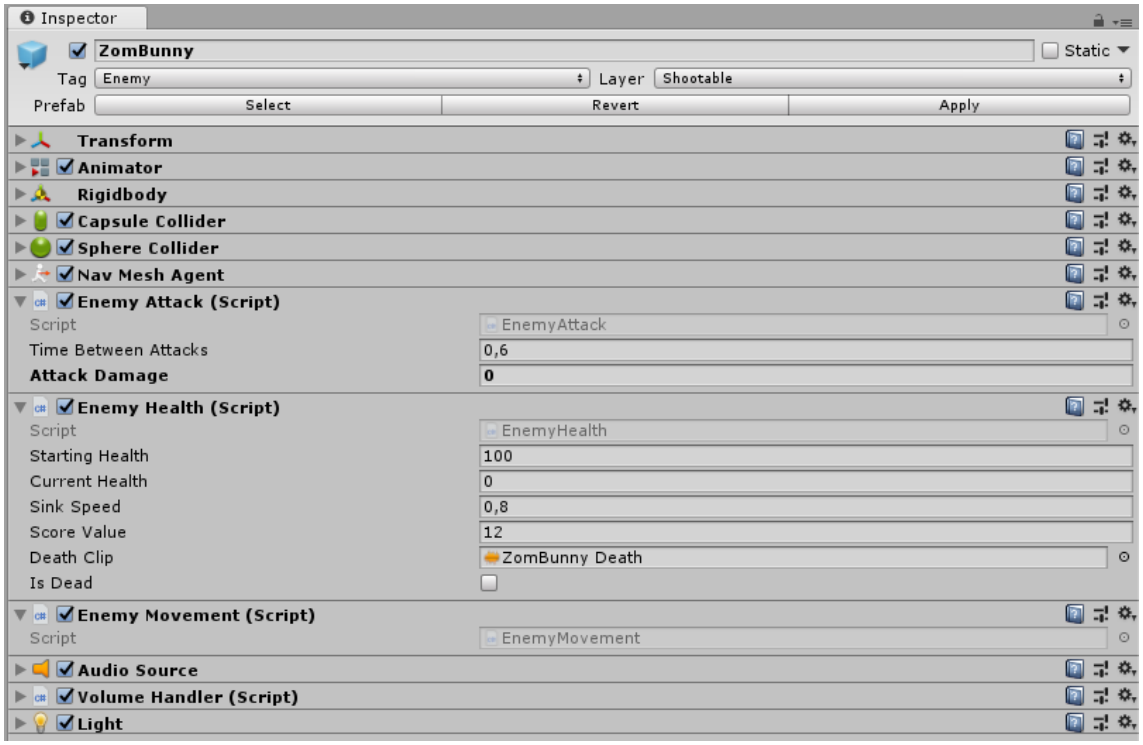
5.3. Implementacja gry

Określenie wysokopoziomowej struktury gry oraz wybór podstawowych growych wzorców projektowych i mechanik o charakterze afektywnym zakończył etap konceptualny tworzenia aplikacji. Kolejnym krokiem było zespolenie pomysłów w spójną całość tak, aby uzyskać gotowy program.

Ze względu na uniwersalność zastosowań, oraz ilość dostępnych materiałów posłużono się bezpłatnym silnikiem gry Unity. Całość aplikacji oparto o rozwiązania przedstawione w jednym z samouczków udostępnionych na stronie producenta oprogramowania. Skorzystano z materiałów filmowych [158] oraz dodatkowego dokumentu [159] omawiającego nowe elementy, które zmieniły się w środowisku Unity od czasu opracowania pierwotnego instruktażu. Umożliwiło to wykorzystanej najnowszej wersji silnika Unity (2018.1.4f1 przeznaczonej dla 64 - bitowych wersji systemów operacyjnych) do wykonania projektu.

Jak wspomniano we wcześniejszych rozdziałach, programowanie z wykorzystaniem Unity opiera się o dwie koncepcje. Pierwsze podejście polega na pisaniu kodu aplikacji w języku C# i dołączaniu go do odpowiednich obiektów (rys. 5.5) w świecie gry. Drugie zaś sprowadza się do interakcji z interfejsem graficznym środowiska Unity. Ułatwia to proces projektowania aplikacji między innymi poprzez umożliwienie podglądu wartości odpowiednich zmiennych. Możliwa jest także zmiana parametrów w czasie kiedy gra jest uruchomiona.

Autorzy samouczka, z którego korzystano, zdecydowali się posługiwać się obiema metodami interakcji z Unity. Jak sami podkreślali było to wynikiem tego, że chcieli przekazać potencjalnemu odbiorcy jak najwięcej przydatnych koncepcji dotyczących tworzenia aplikacji z wykorzystaniem tego oprogramowania. Wpłynęło to na dalszy rozwój gry tworzonej w ramach pracy magisterskiej. Postanowiono korzystać ze schematu zaproponowanego przez prowadzących instruktaż.



Rysunek 5.5. *Skrypty języka C# i inne komponenty środowiska Unity podłączone do obiektu reprezentującego jednego z przeciwników,* źródło: opracowanie własne.

Twórcy poradnika zaproponowali, aby każdą z opracowanych klas przypisać do struktur wyższego poziomu, które mają swoje reprezentacje w postaci jednego bądź kilku obiektów w świecie gry. W ten sposób utworzono dwupoziomową hierarchię. Została ona przedstawiona niżej wraz ze wszystkimi elementami, które zostały zaprogramowane.

1. **Player**, skrypty są związane z głównym bohaterem gry.
   - **PlayerMovement**, opisująca sposób poruszania się postaci. Jej zadaniem jest również aktywowanie animacji ruchu bohatera.
   - **PlayerHealth**, reprezentująca poziom życia postaci. Odpowiada także za integrację z elementem interfejsu symbolizującym aktualny stan tej charakterystyki. Kontroluje też wyświetlanie czerwonego rozbłysku w momencie ataku przeciwników i umożliwia aktywację działań, które mają zostać podjęte po śmierci awatara. Mowa tu wyłączeniu możliwości ruchu i ataku postaci.
   - **PlayerShooting**, odpowiadająca za możliwość strzelania do przeciwników przy pomocy broni palnej. Jej zadaniem jest aktywacja odpowiednich animacji, symbolizujących strzały oraz kontrola fizyki pocisku.

2. **Enemy**, skrypty są związane z przeciwnikami.

   – **EnemyMovement**, opisująca sposób poruszania się przeciwników oraz odpowiadająca za wyznaczanie celu jako aktualne miejsce, gdzie znajduje się gracz.

   – **EnemyHealth**, reprezentująca poziom życia wrogów. Do jej zadań należy zmniejszanie wartości reprezentującej stan zdrowia po otrzymaniu obrażeń w wyniku strzały gracza. Odpowiada również za podjęcie działań po śmierci danego antagonisty, czyli uruchomienie animacji oraz zniszczenie obiektu po zadanym czasie.

   – **EnemyAttack**, odpowiadająca za kontrolę ataku przeciwników. Posiada reprezentację szybkości i mocy z jaką dokonywane są kolejne uderzenia.

3. **Managers**, każdy skrypt jest związany z innym obiektem zarządzającym inną właściwością gry.

   – **ScoreManager**, używana do obliczania ilości punktów zdobytych przez gracza w wyniku pokonania przeciwników. Odpowiada także za zarządzanie tekstem, który jest wyświetlany jako informacja dla użytkownika. W pierwotnej wersji podaje dokładną wartość punktową.

   – **GameOverManager**, odpowiadająca za uruchomienie animacji wskazujących na zakończenie gry.

   – **EnemyManager**, wykorzystywana do kontroli mechanizmu pojawiania się przeciwników. Losowo dobiera miejsca, z którego antagoniści wyłaniają się po określonym czasie z określoną częstotliwością.

4. **Camera**, skrypt związany z widokiem na świat gry.

   – **CameraFollow**, zarządzająca aktualnym widokiem rozgrywki, który jest wyświetlany na ekranie. Tworzy zależność między przedstawianym obrazem a bieżącą pozycją awatara gracza.

Powyższe elementy stanowiły bazę przygotowywanej aplikacji. Kolejnym krokiem w celu implementacji opracowanej struktury gry było wkomponowanie wszystkich wyselekcjonowanych wzorców projektowych. Posiłkowano się przygotowaną strukturą kodu, dokładając nowe funkcje do istniejących rozwiązań lub tworząc nowe klasy i nowe ich zbiory.

W skrypcie PlayerHealth zmieniono sposób zarządzania wyświetlaniem paska aktualnego stanu zdrowia postaci. W pierwotnej wersji poziom tej własności zmieniał się wraz z każdą

modyfikacją jej wartości. Po zmodyfikowaniu aktualizacja odbywała się jedynie wraz z przekroczeniem określonych progów. Pozwoliło to na zaciemnienie obrazu informacji o aktualnym poziomie życia.

Ponadto zaimplementowano tam także funkcje pozwalające na zmienianie szybkości poruszania się przeciwników. Modyfikacja tej wartości następowała co pięć sekund. W wersji bez pętli afektywnej prędkość ruchu wraz z każdą zmianą rosła o zadaną wartość. Sprzężenie zwrotne pozwoliło na dostosowanie poziomu szybkości do aktualnego stanu gracza. Wyliczone wartości, dla obu odmian gry, były stosowane dla wszystkich przeciwników.

$$E_{used} = E_{actual} \cdot \frac{max(t,\ t_{max})}{t_{max}} \tag{5.1}$$

Kolejną klasą, która została zmodyfikowana była PlayerShooting (zobacz: Dodatek A). Dodano elementy wykrywające czas wciśnięcia odpowiedniego przycisku w wersji bez sprzężenia zwrotnego oraz czas zgięcia mięśnia dwugłowego ramienia w wersji z aktywną pętlą afektywną. Jeśli uzyskana wartość przekroczyła określony próg, aktywowano *SuperMoc* zgodnie ze wzorem (5.1). Przyjęto następujące oznaczenia: E_{used} – energia zużyta poprzez aktywację zdolności, E_{actual} – aktualna wartość energii. Następnie korzystając z generatora liczb pseudolosowych określano, ile wrogów ma być natychmiastowo wyeliminowanych, biorąc jako górne ograniczenie minimum z aktualnej liczby przeciwników lub wartość poziomu zużytej energii.

Rysunek 5.6. *Intensywny rozbłysk pomarańczowego światła po użyciu dodatkowej zdolności bohatera,* źródło: opracowanie własne.

Co więcej, dodano również obsługę wskaźnika energii, który działał analogicznie do pierwotnej wersji paska zdrowia. Wskazywał zatem dokładną wartość aktualnej mocy. Przed graczem ukryto jednak sposób, w jaki działa *SuperMoc*. Zaimplementowano także obsługę elementu interfejsu graficznego, który odpowiadał za intensywność rozbłysku światła (rys. 5.6) generowanego użyciem zdolności. Właściwość tę skorelowano z wartością składowej przezroczystości koloru. Do obliczenia jej wartości ponownie wykorzystano stosunek czasu korzystania z *SuperMocy* do czasu maksymalnego. Wraz z każdą kolejną wyświetlaną klatką składowa przezroczystości była pomniejszana o stałą liczbę aż do momentu, gdy wynosiła zero.

Modyfikacja EnemyMovement sprowadziła się do uzupełnienia klasy o metodę umożliwiającą ustawienie parametru szybkości. Z kolei w EnemyHealth dodano możliwość zmniejszania zdrowia przeciwników pod wpływem aktywacji *SuperMocy* oraz gdy dana faza gry dobiegła końca.

Zmiany analogiczne do paska wskazującego poziom życia bohatera, wprowadzono w przypadku wyświetlania ilości punktów. Zastosowano logikę rozmytą z wartościami słownymi takimi jak *TRAGICZNIE*, *KIEPSKO*, *PRZYZWOICIE*, *WSPANIALE*, *FANTASTYCZNIE*. Funkcją przynależności była suma osiągniętych punktów w stosunku do ich liczby potrzebnej do awansowania na kolejny poziom.

Ponadto dodano też funkcję aktywującą ładowanie następnej fazy gry i automatyczną eliminację przeciwników po osiągnięciu wymaganego wyniku.

W oparciu o klasę EnemyManager stworzono analogiczne skrypty o nazwach AffectiveEnemyManager oraz MrNightmareEnemyManager. Oba pozwalały na aktywację dodatkowych miejsc pojawiania się przeciwników. W wypadku tego pierwszego uruchamiały się one po upływie określonego czasu w wersji bez pętli afektywnej lub po wykryciu określonych zależności fizjologicznych u gracza. Jeśli dane warunki nie zostały spełnione, czynność powtarzano po kilku sekundach określoną liczbę razy. MrNightmareEnemyManager zarządzał z kolei aktywacją przeciwników podczas walki z bossem na poziomie czwartym. Zdefiniowano cztery poziomy życia głównego antagonisty, po przekroczeniu których instancjonowani byli pomniejsi przeciwnicy.

Obie klasy umożliwiały aktywację różnych rodzajów przeciwników z jednego gatunku. Aby zaznaczyć odrębność każdego z antagonistów należących do jednej rodziny, zmodyfikowano kolor światła (rys. 5.7), który był przez nich emitowany. Na przykład dla coraz silniejszych wersji przeciwników pierwszego gatunku były to różne odcienie niebieskiego. Czynnikiem różniącym odmienne rodzaje wrogów był czas pomiędzy kolejnymi atakami. Ciemniejszy kolor światła reprezentował mniejszy czas potrzebny do wykonania kolejnego uderzenia.

Rysunek 5.7. *Różne kolory światła dla wrogów tego samego gatunku,*
źródło: opracowanie własne.

W celu realizacji pozostałych założeń dotyczących struktury gry, stworzono pewną liczbę obiektów bez reprezentacji wizualnej. Do każdego z nich przypisano skrypty z klasami odpowiadającymi za inne funkcjonalności. Wyróżniono:

1. **CameraSwitcher**, odpowiadający za zmianę widoku trzecioosobowego na widok pierwszoosobowy.
   - **CameraChange**, wykorzystywana do dezaktywacji widoku izometrycznego, reprezentacji wizualnej gracza oraz głównego sposobu poruszania myszą komputerową. Kolejnym zdaniem skryptu była aktywacja dodatkowego komponentu związanego z obiektem gracza, który generował światło oraz włączenie celownika, który z kolei był zdefiniowany jako element interfejsu graficznego. Istotne było również włączenie nowego sposobu obrotu kamery w oparciu o ruchy myszy komputerowej. Do zadań tej klasy należało również zarządzanie czasem, gdyż zmiana widoku odbywała się cyklicznie po upływie dwustu sekund.
   - **CamMouseLook**, użyta to dekodowania ruchu urządzenia wskazującego na widok kamery. W oparciu o informację o uzyskanym przesunięciu kątowym, rotowano widok w odpowiednią stronę (zobacz: Dodatek A). Istotne okazało się także ograniczenie możliwości ruchów, aby uniemożliwić przenikanie przez podłogę. Konieczne okazało się zastosowanie limitów dolnych i górnych ze względu na sposób w jaki przekazywana była informacja o pozycji kątowej.
2. **AudioObject**, odpowiadający za dźwięk.

- **AudioManager**, jego zadaniem było aktywowanie muzyki odtwarzanej w tle w zależności od aktywowanego poziomu. Upewniono się, że obiekt, do którego podłączono powyższy skrypt jest jedynym takim obiektem w aplikacji poprzez zastosowanie wzorca Singleton.

3. **AlertObject**, zarządzający wyświetlaniem ostrzeżeń i informacji.

   - **ImportantAlertManager**, odpowiadająca za formę oraz czas wyświetlanych komunikatów. Do poprawnego działania potrzebowała referencji do instancji interfejsu graficznego wyświetlającej słowa na ekranie (zobacz: Dodatek A). Unity pozwala na ustawienie tekstu tak, aby jego położenie było względne. W tym przypadku napisy umiejscowiono na środku ekranu. Wszystkie komunikaty wyświetlano czcionką o rozmiarze minimum pięćdziesiąt pikseli i jaskrawozielonym kolorze tak, aby były łatwo dostrzegalne przez gracza. Wspomniana klasa zawierała publiczną metodę pozwalającą na wyświetlanie wymaganych komunikatów przez określony czas. Dzięki temu inne obiekty gry mogły z niej korzystać. Upewniono się, że obiekt, do którego podłączono powyższy skrypt jest jedynym takim obiektem w aplikacji poprzez zastosowanie wzorca Singleton.

4. **LogObject**, odpowiadający za mechanizm zapisywania informacji na temat stanu gry.

   - **LogManager**, jej zadaniem było gromadzenie zdarzeń, które pojawiły się w grze. Zawierała publiczną metodę, która umożliwiała dodawanie różnych instancji do magazynu informacji, który przyjął formę listy. Wspomniana struktura przechowywała obiekty klasy **FmOEvent**, która z kolei zawierała pola do przechowywania czasu, który upłynął od uruchomienia gry i komunikatu zdarzenia. Należy zaznaczyć, że w momencie kończenia pracy programu w Unity wywoływane są metody OnDestroy. Dlatego też właśnie wtedy wszystkie elementy listy zapisywane były do pliku z rozszerzeniem CSV. Co ważne, każde zdarzenie zawierało informację o tym, do której z podgrup wydarzeń należy. Dzięki temu możliwe było późniejsze posortowanie zgromadzonych zdarzeń, a przez to skrócenie czasu analizy danych. Upewniono się również, że obiekt, do którego podłączono powyższy skrypt jest jedynym takim obiektem w aplikacji poprzez zastosowanie wzorca Singleton.

5. **UserObject**, zarządzający danymi pojedynczego użytkownika.

   - **UserManager**, odpowiadała przechowywanie i aktualizowanie informacji na temat gracza. Wśród zbieranych danych można wymienić wprowadzoną nazwę, numer, ilość zdobytych punktów oraz preferencje językowe. Upewniono się, że obiekt, do

którego podłączono powyższy skrypt jest jedynym takim obiektem w aplikacji poprzez zastosowanie wzorca Singleton.

6. **TextObject**, odpowiadający za wyświetlanie tekstów historii. Większość klas podpiętych do tego obiektu miała podobną strukturę zawierającą metody do obsługi przycisków, pól umożliwiających wprowadzenie tekstu oraz wyświetlania odliczenia do rozpoczęcia kolejnego poziomu. Ważnym zadaniem tej klasy było też wyświetlanie napisów na całej powierzchni ekranu w celu dostarczenia użytkownikowi historii związanej z grą oraz informacji o czynnościach, które powinien podjąć. W związku dla każdego ze skryptów omówiono jedynie te informacje, które różniły go od pozostałych.

   – **ManageIntroduction**, wykorzystywany do zarządzania ekranem powitalnym oraz połączeniem z urządzeniem BITalino (r)evolution kit. Uruchomienie komunikacji następowało poprzez odwołanie się do innego obiektu zarządczego tylko wtedy, gdy użytkownik zdecydował się na wybranie wersji gry z pętlą afektywną. Dalsze informacje na ten temat zostaną przedstawione w późniejszych sekcjach niniejszego rozdziału.
   – **ManageHistory**, pozwalał na wyświetlanie różnych elementów historii gry pomiędzy jej kolejnymi poziomami głównymi.
   – **ManageZeroLvl**, dzięki referencji do obiektu LogManager pozwalał na wyświetlanie elementów samouczka w formie alertów. W oparciu o tę mechanikę wyjaśniono w jaki sposób gracz może poruszać się po świecie gry, jak strzelać oraz uwypuklono wszystkie elementy interfejsu graficznego.
   – **ManageEnd**, działanie tego skryptu było analogiczne do ManageHistory. Wyświetlane były informacje o zakończeniu przygody i możliwości zobaczenia najlepszych wyników.
   – **ManageScoreBoard**, umożliwiał odczytanie, wypisanie na ekranie oraz zapisanie do pliku listy wszystkich graczy oraz ich wyników wraz z informacją, o tym, czy używano pętli afektywnej. Po odczytaniu wartości z pliku dane przechowywano w słowniku, którego kluczem był wynik, a wartością lista nazw użytkowników (zobacz: Dodatek A). Zapobiegało to możliwości nadpisania elementu w razie powtórzenia się liczby puntów. Ponadto zastosowanie słownika umożliwiło skonstruowanie listy najlepszych graczy poprzez szybkie posortowanie kluczy.

Po opracowaniu struktury obiektów zapisano je w postaci szkieletów, a następnie dodano do poszczególnych faz rozgrywki. Utworzono hierarchię (rys. 5.8), w skład której wchodziła właściwa część, czyli eliminacja przeciwników oraz fazy, podczas których graczowi przedstawiana

była historia. Pierwsza z nich stanowiła uzupełnienie treści przedstawianych w samouczku [158, 159] o wskazane wyżej elementy. Dodatkowo zmodyfikowano też świat gry poprzez ograniczenie dostępu do miejsc, które generowały błędy związane z poruszaniem się wrogów po planszy.

Rysunek 5.8. *Hierarchia poziomów we Freud Me Out,*
źródło: opracowanie własne.

Dodatkowo, dzięki odpowiedniemu doborowi parametrów, sprawiono, że każdy kolejny poziom był trudniejszy od poprzedniego. Skrócono między innymi czas instancjonowania przeciwników oraz zwiększono ilość punktów potrzebnych, aby przejść do dalszej rozgrywki. Aby gra nie była zbyt trudna, wraz z każdym poziomem zwiększono też nieznacznie ilość początkowego życia głównego bohatera.

Interfejs graficzny służący do przedstawiania fabuły, zarządzania powiadomieniami oparto o fragment instruktażu [158, 159] dotyczący wyświetlania powiadomienia o przegranej rozgrywce. Zmodyfikowano go dodając przyciski oraz pola umożliwiające wpisanie tekstu. Przygotowano też dwie wersje językowe gry.

Ostatnim elementem zmodyfikowanym względem wspomnianego przewodnika [158, 159] było dodanie obsługi klawiszy specjalnych, które umożliwiały między innymi sprawniejsze przeprowadzanie testów.

5.4. Dane Fizjologiczne

Równolegle do wcześniej opisywanego projektu gry stworzono osobną aplikację umożliwiającą akwizycję danych fizjologicznych za pomocą urządzenia BITalino (r)evolution kit. Zgromadzone informacje wykorzystano w celu oddziaływania na strukturę gry przy pomocy zaplanowanych w fazie konceptualnej afektywnych wzorców projektowych. Umożliwiło to późniejsze zamknięcie pętli afektywnej.

W celu integracji sprzętu pomiarowego z platformą Unity pierwotnie skorzystano z rozwiązań dostępnych na stronie producenta [160]. Zamieszony tam interfejs programistyczny był jednak przeznaczony dla znacznie niższej wersji silnika niż ta, którą stosuje się obecnie. Mimo

tego, że Unity potrafi dokonać konwersji projektu do nowszych rozwiązań. skorzystano z innej pracy [161]. Oprócz konwersji kodu do wyższej wersji, twórcy dodali scenę, która umożliwiała podgląd sygnałów fizjologicznych w czasie rzeczywistym. Co istotne, skorzystano też z dołączonej implementacji obliczania tętna na podstawie danych z elektrokardiografu.

Bazując na interfejsie programistycznym i postępując zgodnie ze wskazówkami [162] stworzono obiekt reprezentujący BITalino (r)evolution kit w przestrzeni Unity. Związano z nim następujące skrypty:

1. **BitalinoSerialPort**, dostępny dzięki przygotowanemu interfejsowi programistycznemu. Pozwalał na ustawienie szeregu parametrów związanych z transmisją szeregową zarówno za pośrednictwem uniwersalnej magistrali szeregowej jak i łączności Bluetooth. Najważniejszym parametrem była tutaj nazwa portu, dzięki któremu odbywa się komunikacja.

2. **BitalinoManager**, dostępny dzięki przygotowanemu interfejsowi programistycznemu. Konieczne było między innymi ustawienie prędkości próbkowania. Zdecydowano się na największą możliwą wartość, czyli tysiąc próbek na sekundę. Umożliwiło to uzyskanie wszystkich możliwych danych w celu wychwycenia najsubtelniejszych nawet zmian.

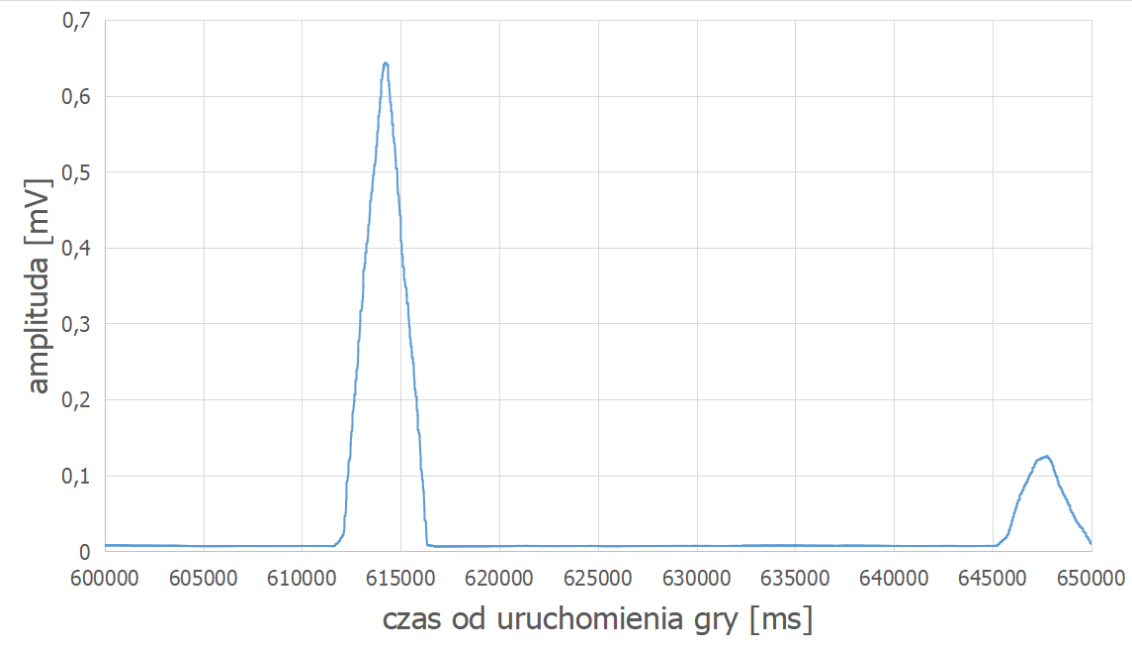
3. **BitalinoReader**, dostępny dzięki przygotowanemu interfejsowi programistycznemu. Dawał możliwość określenia wielkości bufora, do którego zapisywane były odczytane dane. Pozostawiono domyślne ustawienia, czyli dwa tysiące próbek, ze względu na problemy z odczytem tętna, które pojawiały się, gdy wielkość tę próbowano zmniejszać. Z kolei zwiększenie tej wartości powodowało problemy w postaci generowania zbyt dużych opóźnień, które źle wpływały na responsywność systemu.

4. **BitalinoController**, opracowany w ramach niniejszej pracy magisterskiej. Najważniejszą funkcjonalnością tej klasy było wyznaczenie aktualnej, najniższej, najwyżej oraz bazowej średniej wartości sygnałów fizjologicznych (zobacz: Dodatek A). Uzyskane dane dodawano do odpowiednich list wraz z czasami, które upłynęły od uruchomienia gry do ich uzyskania. W momencie niszczenia obiektu dane były zapisywane w osobnym folderze dla każdego gracza. Skrypt zaprojektowano tak, żeby wszystkie wyznaczone wartości były dostępne do podglądu w czasie rzeczywistym.

Posługując się opracowaniem [161] wyznaczono wartości tętna w oparciu o dane z elektrokardiografu. Znaleziono lokalne maksima reprezentujące najwyższą wartość piku w zespole QRS. Za ekstremum przyjęto wartość z ciągu danych rosnących przekraczającą średnią sygnału pomnożonego przez $\frac{3}{2}$. Ponadto, aby daną można było uznać za lokalne maksimum, następna otrzymana po niej wartość musiała być początkiem szeregu malejącego. Różnicę pomiędzy zespołami QRS odnoszono do prędkości próbkowania. W ten sposób obliczano wartość uderzeń

na sekundę dla całego ciągu próbek pochodzących z bufora, a następnie liczono średnią arytmetyczną ze wszystkich uzyskanych wyników.

Przed przystąpieniem do wyznaczenia charakterystyki ilości uderzeń serca na sekundę, sygnał otrzymany z BITalino przepuszczono przez filtr pasmoprzepustowy. Wyeliminowano częstotliwości wyższe oraz niższe niż zadany poziom. Pasma, o których mowa związane są z oddychaniem osoby podłączonej do sprzętu badawczego. Za artykułem [163] przyjęto ograniczenie dolne wynoszące 5Hz, oraz dolne jako 0.67Hz.

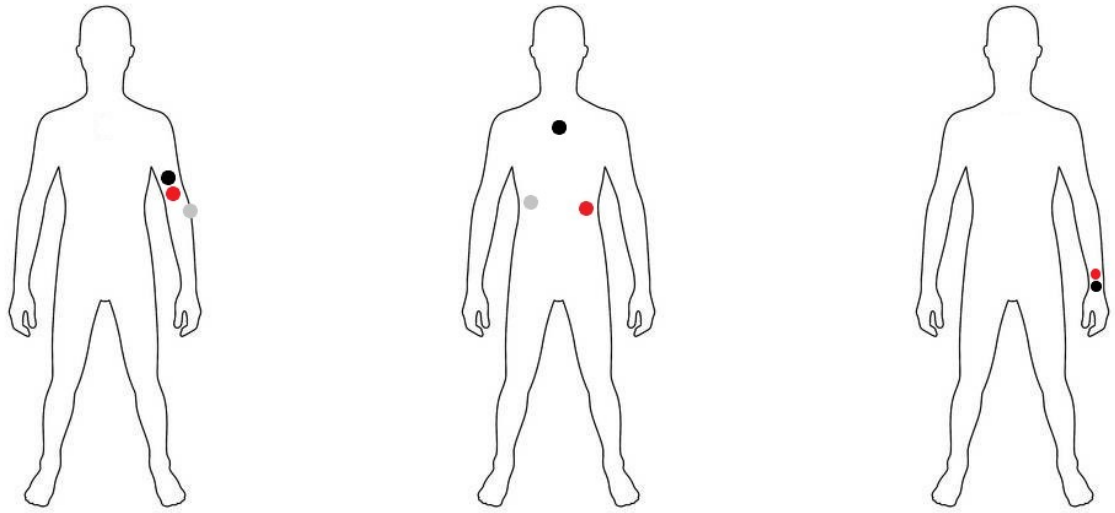


Rysunek 5.9. *Sygnał z elektromiografu po odfiltrowaniu szumu,*
źródło: opracowanie własne.

W przypadku sygnału związanego z pracą mięśni zauważono, że co druga próbka to bardzo duże i niepożądane wartości. W związku z tym sygnał oscylował. Po odfiltrowaniu wartości wyższych niż 3 jednostki, sygnał stał się wyraźnie gładszy (rys. 5.9). Z otrzymanych serii próbek wyznaczano jedną wartość średnią.

W przypadku reakcji elektrodermalnej zastosowanie filtrowania nie okazało się konieczne. W związku z tym dla serii dwóch tysięcy próbek wyznaczano chwilową wartość średnią.

Istotne okazało się także umiejscowienie elektrod umożliwiających pomiary (rys. 5.10). W przypadku elektrokardiografu skorzystano z rozwiązania, którego użyto już w podobnych eksperymentach [21, 84] z platformą BITalino (r)evolution kit. Elektrodę odniesienia dla elektromiografu umieszczono na wystającej kości łokcia, zaś dwie pozostałe na docelowym mięśniu dwugłowym ramienia [164].



(a) EMG (b) EKG (c) EDA

Rysunek 5.10. *Sposoby przypięcia elektrod dla różnych modalności, kolor czerwony oznacza elektrodę dodatnią, czarny ujemną, a szary uziemienie*
źródło: opracowanie własne

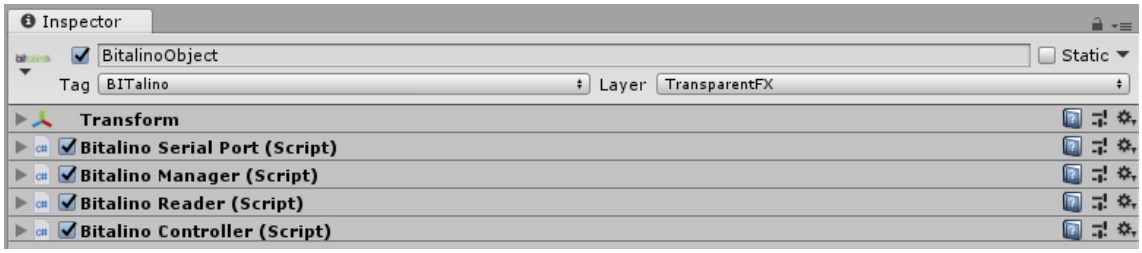
W przypadku reakcji elektrodermalnej wypróbowano kilka najbardziej skorelowanych z wykrywaniem emocji [165] miejsc przyłożenia elektrod. Okazało się jednak, że dla elektrod, które nie były suche, otrzymywany sygnał był wrażliwy na ruch albo na skurcze mięśni rejestrowane tym samym urządzeniem. W związku z tym, że docelowo pomiary miały być stosowane do gry, wyeliminowano palce rąk jako miejsca przyłożenia elektrod mimo, że cechowały się najlepszymi wartościami korelacji. Ostatecznie elektrody umieszczono poniżej nadgarstka po wewnętrznej stronie ręki, tak jak to ma miejsce w przypadku niektórych opasek pomiarowych [84].

5.5. Zamknięcie pętli afektywnej

Aby gra realizowała w pełni założenia programowania afektywnego przedstawione przez Picard [9] potrzebne było domknięcie pętli afektywnej. Oznacza to, że pewne elementy gry musiały zmieniać się w oparciu o wykryte stany emocjonalne przy pomocy sygnałów fizjologicznych.

Zintegrowanie dwóch aplikacji w całość było relatywnie prostym zadaniem, gdyż polegało na przeniesieniu opisanego wyżej mechanizmu (rys. 5.11) akwizycji danych pomiarowych do struktury gry komputerowej. Upewniono się również, że wskazany obiekt jest jedynym takim obiektem w aplikacji poprzez zastosowanie wzorca Singleton.

ManageIntroduction, czyli wspomniany wyżej skrypt zarządzający wprowadzeniem odwoływał się do jednej z metod z klasy BitalinoController, która aktywowała rozpoczęcie odczytu danych oraz po odczekaniu kilku sekund, kalibrację. Pominięcie pierwszych danych pomiarowych miało na celu odrzucenie wartości zaszumionych przez start urządzenia. W czasie kalibracji wyznaczana była średnia wartość bazowa dla wszystkich sygnałów osobno. Równocześnie, na ekranie wyświetlano prośbę, aby użytkownik wyprostował się, zaprzestał ruchów oraz ułożył swoje ręce tak, jak do grania.



Rysunek 5.11. *BitalinoObject wraz ze wszystkimi potrzebnymi skryptami umieszczony w świecie gry,* źródło: opracowanie własne.

Po procesie kalibracji wartości bazowych, następował regularny pomiar wszystkich sygnałów fizjologicznych. Razem z postępem gry, wyznaczano też wartości maksymalne i minimalne sygnałów fizjologicznych, odrzucając wartości skrajne. W przypadku tętna było to ± 15 uderzeń serca na minutę od wartości bazowej, w przypadku reakcji elektrodermalnej ± 0.2 mikrosimensów od poziomu bazowego. W przypadku elektromiografu nie stosowano obostrzeń. Zastosowanie wspomnianych ograniczeń miało na celu zmniejszenie szerokości zmienności mierzonych sygnałów, a nie ich filtrowanie.

Dla każdej z mechanik afektywnych opracowano osobne reguły i funkcje, które pozwoliły na ich aktywacje. Przyjęto następujące oznaczenia: HR_{max} – maksymalna wartość tętna, HR_{min} – minimalna wartość tętna, HR_{av} – średnia chwilowa wartość tętna, EDA_{max} – maksymalna wartość reakcji elektrodermalnej, EDA_{min} – minimalna wartość reakcji elektrodermalnej, EDA_{av} – średnia chwilowa wartość reakcji elektrodermalnej, EMG_{av} – średnia chwilowa wartość pochodząca z elektromiografu oraz EMG_{calib} – bazowa wartość pochodząca z elektromiografu. Wyróżniono tutaj:

1. **Dostosowanie szybkości poruszania się przeciwników pod wpływem poziomu uderzeń serca na minutę.** W tym wypadku skorzystano z właściwości funkcji logistycznej, która pozwalała na subtelne przejście od jednej wartości do drugiej. Współczynniki funkcji (5.2) dobrano tak, żeby zwracała wartości z przedziału od 2 do 7, gdzie punkt środkowy wyznaczany jest jako średnia arytmetyczna największej i najmniejszej wartości tętna. Współczynnik stromości $k = 0.3$ dobrano poprzez analizę eksperymentalną tak,

żeby uzyskać jak największe zróżnicowanie dla danych z przedziału od 50 do 90.

$$2 + \frac{5}{1 + e^{\,0.3 \cdot (x - \frac{HR_{max} + HR_{min}}{2})}} \tag{5.2}$$

2. **Aktywacja dodatkowych miejsc pojawiania się przeciwników pod wpływem poziomu uderzeń serca na minutę oraz poziomu reakcji elektrodermalnej**. Włączenie tej mechaniki następowało jeśli przez pięć kolejnych sekund spełniona była formuła (5.3). Jeśli warunek nie został spełniony, rozpoczynano ponowne sprawdzanie. Całość powtarzano maksymalnie pięć razy razy.

$$HR_{av} < \frac{HR_{max} + HR_{min}}{2} \quad \wedge \quad EDA_{av} < EDA_{max} \tag{5.3}$$

3. **Dostosowanie losowości miejsc pojawiania się przeciwników pod wpływem poziomu uderzeń serca na minutę oraz poziomu reakcji elektrodermalnej**. Skorzystano z logiki rozmytej. Wartości sygnałów fizjologicznych pozwoliły na określenie aktualnego stanu pobudzenia gracza. Wyznaczono trzy poziomy: WYSOKIE, STANDARDOWE, NISKIE. O przynależności do każdego ze stanów decydowało spełnienie jednej z poniższych formuł (5.4), (5.5), (5.6).

$$HR_{av} \geq \frac{HR_{max} + HR_{min}}{2} \wedge EDA_{av} \geq \frac{EDA_{max} + EDA_{min}}{2} \Rightarrow \text{WYSOKIE} \Rightarrow 0 \tag{5.4}$$

$$HR_{av} \leq \frac{HR_{max} + HR_{min}}{2} \wedge EDA_{av} \leq \frac{EDA_{max} + EDA_{min}}{2} \Rightarrow \text{NISKIE} \Rightarrow 5 \tag{5.5}$$

$$\text{W INNYM WYPADKU} \Rightarrow \text{STANDARDOWE} \Rightarrow 2 \tag{5.6}$$

4. **Użycie *SuperMocy* poprzez napięcie mięśnia dwugłowego ramienia**. Spełnienie formuły (5.7) sprawiało, że uruchamiano stoper, który odmierzał czas, przez który warunek był spełniony.

$$EMG_{av} \geq 3 \cdot EEG_{calib} \tag{5.7}$$

Implementacja wyżej wymienionych wzorców projektowych pozwoliła na stworzenie gry, która realizowała zagadnienia programowania afektywnego. Sygnały fizjologiczne umożliwiały oddziaływanie na model gry, a informacje pochodzące z rozgrywki w założeniu mogły wpływać na stan emocjonalny gracza.

6. Ewaluacja

6.1. Procedura eksperymentalna

Przeprowadzone badanie składało się z trzech głównych faz. Podczas pierwszej części osoba poddawana eksperymentowi grała w przygotowaną grę komputerową w wersji z pętlą afektywną, czyli z nałożonymi urządzeniami pomiarowymi. Kolejny wariant był analogiczny, stosowano jednak opcję bez pętli afektywnej. Następnie wypełniano ankietę.

Aby ujednolicić przeprowadzone testy, sporządzono instrukcję dla eksperymentatora wypisując najważniejsze elementy procedury badawczej. Przyjęto oznaczenie OB jako skrót wyrażenia *Osoba Badana*. Wyróżnione zostały:

1. Przywitanie OB.
2. Wskazanie stanowiska, przy którym OB powinna usiąść.
3. Wręczenie OB krótkiej instrukcji. Znajduje się w Dodatku B.
4. Odpowiedzenie na ewentualne pytania OB.
5. Pomoc OB w nałożeniu elektrod na wskazane miejsca ciała.
6. Uruchomienie aplikacji w środowisku Unity.
7. Wpisanie nazwy oraz identyfikatora liczbowego OB.
8. Sprawdzenie, czy grze udało się nawiązać połączenie z platformą BITalino (r)evolution kit. W razie problemów z komunikacją, ponowienie próby.
9. Pilnowanie czasu gry. Około 15 minut.
10. Pomoc w usunięciu elektrod z ciała.
11. Umożliwienie przerwy na prośbę OB.

12. Wpisanie identycznej nazwy oraz identyfikatora liczbowego zwiększonego o 1 w porównaniu do poprzedniego numeru.
13. Pilnowanie czasu gry. Około 15 minut.
14. Wręczenie OB kwestionariusza ewaluacyjnego. Znajduje się w Dodatku C.
15. Wyjaśnienie problematycznych kwestii, odpowiedź na pytania OB.
16. Wręczenie OB ewentualnego upominku.
17. Podziękowanie OB za udział w badaniu.

W procedurze nie uwzględniono planów awaryjnych ze względu na to, że oprócz ustanowienia łączności z BITalino (r)evolution kit nie zidentyfikowano problematycznych kwestii.

6.2. Sprzęt i oprogramowanie

Testy odbyły się z wykorzystaniem urządzenia pomiarowego BITalino (r)evolution kit. Zastosowano także komputer osobisty Xiaomi Mi Notebook 13. Cechował się on następującą konfiguracją sprzętową:

- Procesor: Intel Core i5-6200U.
- Karta graficzna: NVIDIA GeForce 940MX.
- Pamięć: 8192MB, DDR4 2133MHz, pojedynczy kanał.
- Dysk: Samsung PM951 NVMe MZVLV256, 256GB.
- Wielkość matrycy: 13.3 cale.
- Rozdzielczość ekranu: 1920x1080.

Odnosząc się do oprogramowania, użyto:

- System operacyjny: Windows 10 w wersji 1803.
- Środowisko programistyczne: Unity w wersji 2018.1.4f1.
- Gra: Freud Me Out w wersji 1.0.

6.3. Uczestnicy badania

Przebadanych zostało 9 osób, z czego 2 to kobiety a 7 to mężczyźni. W związku z tak dużą dysproporcją, niemożliwe okazało się dokonanie analizy wyników ze względu na płeć.

Uczestnicy eksperymentu należeli do grupy studentów, osób pracujących lub uczniów szkół ponadgimnazjalnych. Wiek osób badanych wahał się od 16 do 24 lat.

6.4. Zgromadzone informacje

Pozyskane dane miały zróżnicowany charakter. Wyróżniono więc trzy następujące grupy:

– Próbki sygnałów z elektrokardiografu, elektromiografu oraz reakcji elektrodermalnej mierzone przy pomocy urządzenia BITalino (r)evolution kit. Zapisywano obliczone wartości średnie, maksymalne, minimalne oraz dane po kalibracji ze względu na to, że to właśnie one były wykorzystywane przy aktywacji afektywnych growych wzorców projektowych. Każda z wymienionych wartości została także złączona z czasem od uruchomienia gry, kiedy ją uzyskano.

– Logi zdarzeń występujących podczas grania w grę komputerową. Każdy log został także złączony z czasem od uruchomienia gry, kiedy nastąpiło zdarzenie.

– Kwestionariusze wypełnione po zakończonej rozgrywce.

6.5. Analiza wyników

Przeprowadzono interpretację uzyskanych informacji w oparciu o dane pochodzące z kwestionariuszy oraz logi zapisane podczas gry. Celowo pominięto analizę danych, które pochodziły bezpośrednio z urządzenia pomiarowego, gdyż założenia pracy nie obejmowały ustalenia dokładności uzyskiwanych sygnałów fizjologicznych. Na podstawie [84] założono, że otrzymane sygnały fizjologiczne cechują się wystarczającą dokładnością.

Jako główny cel analizy przyjęto określenie jak zaimplementowane we Freud Me Out afektywne growe wzorce projektowe wpływają na odbiór gry przez użytkowników. Istotne było także ustalenie jak mechaniki wykorzystujące dane fizjologiczne wpływają na zmienność stanów rozgrywki poprzez obserwację odpowiednich parametrów.

Pytania w kwestionariuszu podzielono na cztery części. Poniżej pogrupowano wszystkie kwestie stosując kolejność jaką użyto w ankiecie. Zagadnienia dotyczyły:

1. Odbioru trudności obu wersji gry.

2. Negatywnego wpływu wywołanego przez sprzęt pomiarowy.

3. Subiektywnego odczucia, które afektywne growe wzorce projektowe wpływały pozytywnie na pobudzenie.

4. Określenia, czy lepsze były mechaniki wykorzystujące dane fizjologiczne, czy ich nieafektywne odpowiedniki.

Pierwsze, co można zauważyć po zestawieniu wyników to wysoka zgodność odpowiedzi pomiędzy różnymi osobami badanymi. Skutkuje to ułatwieniem interpretacji danych oraz możliwością skutecznego wnioskowania. Wyniki zebrano w tabele i w dalszej kolejności omówiono. We wszystkich wykazach, komórki zawierające wartości numeryczne symbolizują liczbę uczestników eksperymentu, którzy w kwestionariuszu zagłosowali za daną opcję.

Tabela 6.1. Trudność gry.

	z pętlą afektywną	**bez pętli afektywnej**
łatwy	2	0
pomiędzy łatwym a średnim	4	2
średni	3	0
pomiędzy średnim a trudnym	0	3
trudny	0	4

Obserwując wyniki (tab. 6.1) dotyczące oceny poziomu gry, można zauważyć wyraźne przesunięcie rozkładu rezultatów. Wszystkie odpowiedzi dla wersji gry z pętlą afektywną skupiały się wokół trzech pierwszych wartości, sugerując poziom łatwy lub średni. Zdecydowana większość uczestników badania zagłosowała za tym, że gra bez mechanik operujących na danych fizjologicznych jest trudna. Może to sugerować, że pętla afektywna pozwala dostosować poziom gry do aktualnych umiejętności oraz stanu użytkownika.

Tabela 6.2. Wpływ urządzeń pomiarowych na możliwość gry.

czy sprzęt pomiarowy przeszkadzał?	
nie	2
raczej nie	6
trudno powiedzieć	1
raczej tak	0
tak	0

Żadna z przebadanych osób badanych nie odpowiedziała twierdząco na pytanie, czy sprzęt do pomiaru sygnałów fizjologicznych przeszkadzał jej w grze (tab. 6.2). Tylko jeden uczestnik eksperymentu nie był w stanie stwierdzić wpływu BITalino (r)evolution kit. Tak duża przewaga głosów za tym, że stosowanie wybranego urządzenia nie niesie za sobą negatywnych konsekwencji może sugerować, że wskazanie tego sprzętu jako platformy akwizycyjnej dla programowania afektywnego jest uzasadnione i właściwe.

Tabela 6.3. Wpływ mechanik na pobudzenie.

	czy zwrócono uwagę?	**czy miało znaczenie pobudzające?**
przedstawiana historia	9	6
ukrycie informacji o dokładnej liczbie punktów	9	8
ukrycie informacji o dokładnej liczbie poziomu życia	0	0
rozwój bohatera - pojawienie się *SuperMocy*	9	6
pojawianie się ostrzeżeń/informacji na ekranie	9	7
zmiana widoku z oddali na widok pierwszoosobowy	4	4
wrogowie i walka	9	9
przeszkody i ich omijanie	6	2
pojawienie się nowych rodzajów wrogów	9	7
losowe miejsca pojawiania się przeciwników	3	3
aktywacja dodatkowych miejsc pojawiania się przeciwników	3	2
zmiana szybkości poruszania się przeciwników	9	9
możliwość sprawdzenia i porównania swojego wyniku z innymi graczami	9	6

Wszystkie osoby zauważyły wzorce przedstawianej historii, ukrycia informacji o dokładnej liczbie punktów, rozwoju bohatera, pojawiania się ostrzeżeń na ekranie, wrogów i walki, pojawienia się nowych rodzajów wrogów, zmiany szybkości poruszania się przeciwników oraz możliwości sprawdzenia i porównania swojego wyniku z innymi graczami. Każdy z nich został oceniony jako potencjalnie redukujący znudzenie i/lub zrelaksowanie przez większość osób. A zatem subiektywna ocena (tab. 6.3) graczy sugeruje, że każda z wymienionych mechanik miała znaczenie afektywne.

Nie wszyscy gracze zwrócili uwagę na aktywację dodatkowych miejsc pojawiania się przeciwników, mimo zastosowania alertów informujących o tym zdarzeniu. Jako powód można tutaj

podać to, że gracze byli zbyt zajęci interakcją z przeciwnikami i nie zwrócili uwagi na dodatkowe miejsca instancjonowania wrogów. Żaden z graczy nie dostrzegł wzorca ukrycia informacji o dokładnym poziomie życia bohatera. Może to sugerować, że użytkownicy nie kontrolowali tego paska, gdyż byli zajęci intensywną walką. Nie dziwi, że tylko cztery osoby dostrzegły zmianę kamery. Jest to związane z tym, że nie wszyscy uczestnicy dotarli do właściwego poziomu, kiedy ta mechanika była aktywowana.

Tabela 6.4. Ulubione mechaniki.

	z pętlą afektywną	**bez pętli afektywnej**
użycie *SuperMocy*	8	0
aktywacja dodatkowych miejsc pojawiania się przeciwników	2	0
zmiana szybkości przeciwników	6	1
losowość miejsc pojawiania się przeciwników	1	0

Tabela 6.5. Generowanie immersji.

	z pętlą afektywną	**bez pętli afektywnej**	**nie dostrzeżono różnicy**
użycie *SuperMocy*	9	0	0
aktywacja dodatkowych miejsc pojawiania się przeciwników	3	0	6
zmiana szybkości przeciwników	6	0	3
losowość miejsc pojawiania się przeciwników	0	0	9

Tabela 6.6. Dostosowanie poziomu gry.

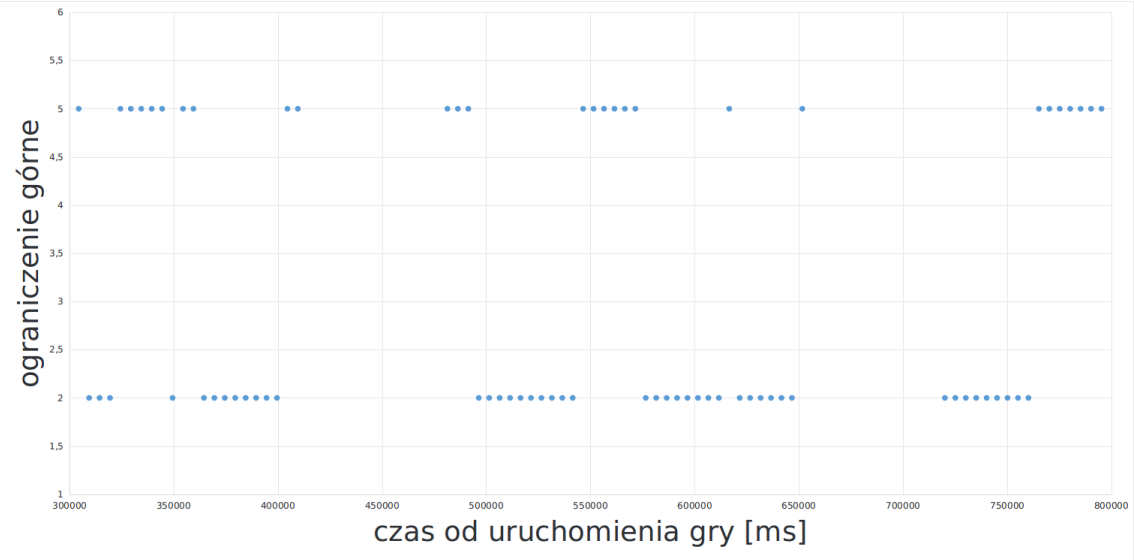
	z pętlą afektywną	**bez pętli afektywnej**	**nie dostrzeżono różnicy**
użycie *SuperMocy*	4	2	3
aktywacja dodatkowych miejsc pojawiania się przeciwników	3	0	6
zmiana szybkości przeciwników	8	0	1
losowość miejsc pojawiania się przeciwników	0	0	9

Po dokonaniu analizy wyników (tab. 6.4, 6.5, 6.6) dotyczących struktur gry opartych o dane fizjologiczne nie dziwi, że jako ulubioną mechanikę wskazano tę, która umożliwiała aktywowanie specjalnej umiejętności bohatera za pomocą zgięcia mięśnia dwugłowego ramienia. Podczas

eksperymentu obserwowano uśmiechnięte twarze uczestników, którzy korzystali z tej zdolności. Dodatkowo również tę mechanikę wskazano jako pozwalającą na wygenerowanie immersji poprzez umożliwienie zespolenia ze światem gry. Jest to prawdopodobnie wynikiem tego, że jej działanie było oparte o bezpośrednie wykorzystanie sygnałów fizjologicznych.

Wyciągnięcie bardzo ciekawych wniosków umożliwia również obserwacja wzorca odpowiedzialnego za regulację szybkości przeciwników. Został on wskazany jako drugie w kolejności najciekawsze rozwiązanie. Ponadto najwięcej osób wskazało go jako ten, który pozwolił na dostosowanie poziomu rozgrywki do umiejętności graczy. Odwołując się do wcześniejszych danych (tab. 6.1), można stwierdzić, że to prawdopodobnie właśnie ten element odpowiadał za to, że wersja gry z pętlą afektywną była odczuwana jako łatwiejsza niż ta bez sprzężenia zwrotnego.

W związku z tym, co pokazano już wcześniej, że niektóre z mechanik zostały niezauważone, nie można mieć stuprocentowej pewności co do tego, który element tak naprawdę ułatwiał rozgrywkę. Po złączeniu danych z kwestionariuszy i logów, widać wyraźnie, że osoby, dla których wzorzec aktywacji nowych miejsc pojawiania się przeciwników nie uruchamiał się, wskazywały właśnie jego jako ten, który wpływa na dostosowanie poziomu gry. Co więcej, ci trzej gracze zagłosowali też za tym, że właśnie ta mechanika była odpowiedzialna za umożliwienie *wczucia się* w świat gry.



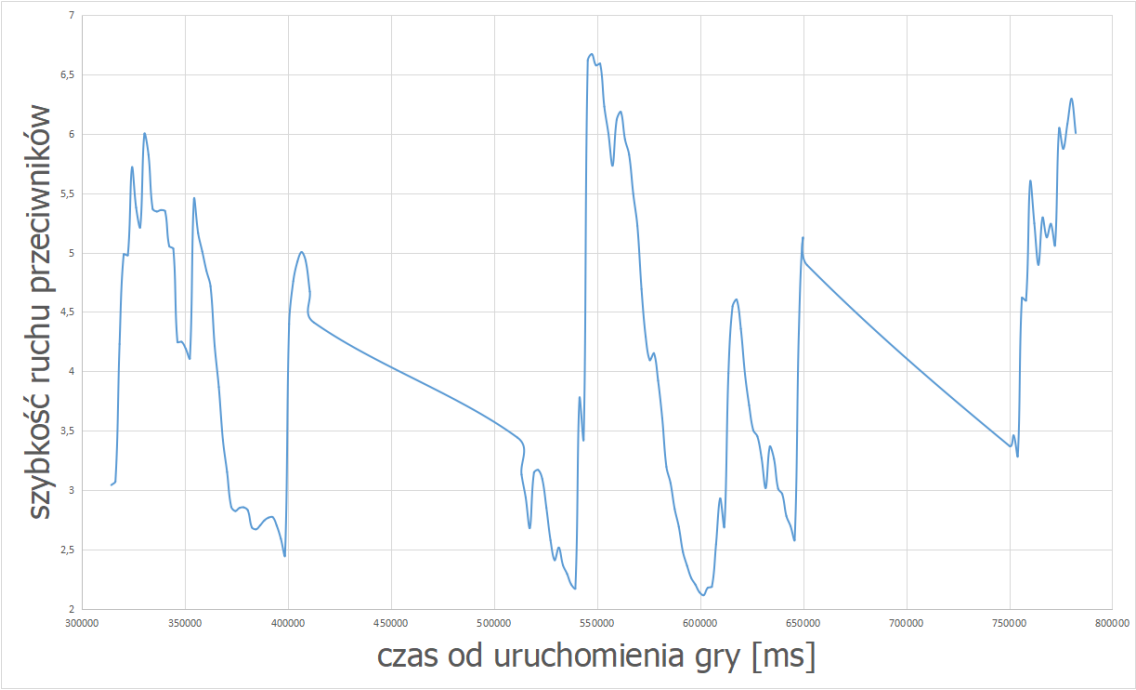
Rysunek 6.1. *Zmienność ograniczenia losowości pod wpływem stanu afektywnego na przykładzie jednego z badanych,* źródło: opracowanie własne.

W przypadku analizy ograniczenia losowości (rys. 6.1) pojawiania się miejsc przeciwników można zauważyć, że żaden gracz nie doświadczył jej na poziomie świadomości (tab. 6.6). Nie

jest to jednak zaskakujące, gdyż gra wymagała dużego skupienia i wychwycenie takiej subtelności jak wspomniana mechanika było trudnym zadaniem. Ponownie można odwołać się tutaj do logów i wskazać, że w przypadku większości graczy wzorzec ten został poprawnie aktywowany.

Co jednak najbardziej znamienne, tylko 3 na 88 razy, we wspomnianej wyżej serii porównań, zagłosowano za wyższością mechanik niewykorzystujących danych fizjologicznych. Wnioski jakie można wysnuć z tej analizy napawają optymizmem w kontekście tworzenia aplikacji afektywnych. Oznacza to bowiem, że użytkownicy preferują programy, w których dane fizjologiczne umożliwiają interakcję ze światem gry.

Analiza logów pozwoliła też sprawdzić, która wersja mechaniki *SuperMocy* była chętniej wykorzystywana. Ilość aktywacji wyniosła odpowiednio 32 oraz 16, kolejno dla wersji ze sprzężeniem zwrotnym i bez sprzężenia zwrotnego. Mogłoby to sugerować, że uczestnicy badania woleli korzystać z *SuperMocy* opartej o aktywację skurczeniem mięśnia dwugłowego ramienia. Ta statystyka może być jednak niewłaściwa do oceny tego, którą mechanikę preferowano. Powodem jest to, że wszyscy gracze w pierwszej kolejności, zgodnie z procedurą, korzystali z wersji gry, której działanie było oparte o dane fizjologiczne.



Rysunek 6.2. *Zmienność szybkości ruchu przeciwników pod wpływem zmian pracy serca na przykładzie jednego z badanych,* źródło: opracowanie własne.

Bardzo dobrze prezentują się również rezultaty, które osiągnięto poprzez wykorzystanie funkcji logistycznej do obliczenia preferowanej prędkości przeciwników (rys. 6.2). Okazało się,

że w przypadku każdej osoby osiągnięto zróżnicowane wartości szybkości ruchu w zakresie od 3.5 do 6.5 jednostek. Dodatkowo, w przypadku 5 osób osiągnięto prawie pełne zakresy – od wartości zbliżających się do 2 do liczb bliskich 7.

Co warte odnotowania, żadna z osób badanych nie zdecydowała się na wypełnienie pola przeznaczonego na dodatkowe uwagi. Jednakże prawie wszyscy po zakończeniu eksperymentu dzielili się swoimi spostrzeżeniami na temat struktury gry. Wskazywano elementy wymagające poprawy bądź reagowano entuzjazmem samą możliwość zagrania we Freud Me Out.

7. Podsumowanie

7.1. Wnioski

Niniejsza praca miała na celu odwołanie się do procesu budowania gier elektronicznych na platformie Unity. Realizując postawione zadanie skorzystano z idei growych wzorców projektowych.

Wykonanie projektu rozpoczęto od obszernego przeglądu literaturowego z zakresu interakcji pomiędzy ludźmi a komputerami. Skupiono się na nowych technikach komunikacji z urządzeniami. Odwołano się do możliwości oferowanych przez wirtualną rzeczywistość, mobilnych asystentów oraz systemów wykorzystujących modelowanie emocji. Przyłożono dużą wagę do paradygmatu programowania afektywnego i jego głównego konceptu, czyli pętli afektywnej.

Przeprowadzono badania na gruncie tworzenia gier komputerowych, dokonując analizy współcześnie dostępnych silników umożliwiających tworzenie tych aplikacji. Przedstawiono również sam proces projektowania gier, które oddziałują na ludzkie emocje.

Wskazano sposoby monitorowania oraz analizy stanu graczy. Wyróżniono metody bezpośrednie, oparte na sygnałach fizjologicznych i pośrednie, operujące na metrykach behawioralnych. Pokazano też, w jaki sposób modelowane są stany emocjonalne.

Część projektową rozpoczęto od selekcji growych wzorców projektowych, umożliwiających stworzenie koncepcji struktury gry. Następnie opracowano własne mechaniki, które opierały swoje działanie o dane fizjologiczne.

Zaimplementowano główny element systemu, czyli grę komputerową. Projekt uzupełniono o obsługę BITalino (r)evolution kit, które umożliwiało akwizycję danych fizjologicznych. W ten sposób zamknięto sprzężenie zwrotne, umożliwiając generowanie wzajemnego wpływu pomiędzy użytkownikiem a programem komputerowym.

Stworzone rozwiązanie następnie przetestowano w celu ewaluacji systemu. Porównano dwie wersje gry, jedną wykorzystującą pętlę afektywną i drugą, która nie zawierała sprzężenia zwrotnego. Ocenę aplikacji oparto o logi zdarzeń z gry oraz o dane dostarczone przez użytkowników wypełniających specjalny kwestionariusz po zakończeniu rozgrywki.

Odnosząc się do uzyskanych wyników, można stwierdzić, że program zaprezentowany w niniejszej pracy magisterskiej działa poprawnie i realizuje założenia przyświecające projektowi. Za duży sukces należy uznać to, że gracze zaproszeni do udziału w badaniach znacznie częściej wskazywali aplikację wykorzystującą afektywne wzorce projektowe jako pozwalającą na pełniejsze obcowanie ze światem gry.

Daje to nadzieję na możliwość opracowania znacznie bardziej złożonych rozwiązań i rozpowszechnienie idei programowania bazującego na emocjach użytkownika wśród przedstawicieli branży komercyjnych gier elektronicznych.

7.2. Propozycje przyszłych prac

Należy zauważyć, że interpretacja wyników ankiet stanowiła pierwszą fazę ewaluacyjną. Rozwinięciem projektu będzie szczegółowa analiza statystyczna zebranych danych fizjologicznych i próba skorelowania ich z afektywnymi wzorcami projektowymi. W wyniku tych prac przewidziano powstanie artykułu naukowego.

Istnieje też wiele możliwości rozwoju stworzonego systemu. Ważne jest, aby dalsze plany zacząć od elementów bezpośrednio łączących się z założeniami pracy. Należy wyselekcjonować bądź stworzyć nowe schematy, które mogą mieć znaczenie afektywne i dołożyć te funkcjonalności do świata opracowanej gry.

Przykładem niech będzie wzorzec **długoterminowych konsekwencji wcześniejszych wyborów**. Można wyobrazić sobie sytuację, w której osoba przechodzi przez całą grę, ale nigdy nie używała umiejętności *SuperMocy*. Wykrycie takiego stanu, powinno w tym wypadku aktywować dodatkowe możliwości, niedostępne dla wszystkich innych użytkowników.

Inną drogą rozwoju, którą można obrać jest wykorzystanie heurystyki wywoływania emocji. Jest ona znacznie mniej ścisła niż użyte w niniejszej pracy funkcje pomocy oraz utrudniania przebiegu gry.

Sama aplikacja generuje także kilka elementów, które wymagają poprawy. Istotne jest, aby podczas zmiany prędkości operować nie tylko wartości szybkości liniowej przeciwników, ale także zwiększać charakterystykę związaną z ruchem obrotowym. Co więcej, również algorytm, z którego skorzystano w celu wyznaczenia najkrótszej ścieżki pomiędzy awatarem gracza a antagonistami powinien zostać ulepszony.

Ponadto, należy rozważyć też opracowanie rozwiązań pomagających w dostosowywaniu ekranu aplikacji poprzez wsparcie rozdzielczości innych niż natywne 1920x1080.

W kontekście detekcji stanów afektywnych, przydatne byłoby opracowanie metod pozwalających na uwzględnienie zmienności rytmu serca. Jest ono zależne od wielu parametrów takich

jak wiek, zażywane leki i stosowane używki. Generuje to kolejny problem, czyli względnie tych parametrów w celu obliczenia wspomnianej charakterystyki.

Jako rozwiązanie można zaproponować dodanie kolejnych zmiennych, pozwalając użytkownikowi na podanie odpowiednich informacji przed uruchomieniem gry. Z drugiej strony, aby uprościć ten proces warto rozważyć obsługę odczytu pożądanych informacji z kont portali społecznościowych takich jak Faceboook. Warto też zastanowić się, co z anonimizacją takich informacji i jak wyglądałoby to w świetle ogólnego rozporządzenia o ochronie danych.

Udoskonalenia wymaga także metodologiczna strona prowadzonych badań. Jak wskazano we wcześniejszym rozdziale, struktura procedury była niezmienna względem każdego uczestnika eksperymentu. Najpierw należało grć w grę z pętlą afektywną, a następnie uruchomić aplikację bez sprzężenia zwrotnego. Z metodologicznego punktu widzenia należy dołożyć proces randomizacji przypisywania badanych do grupy kontrolnej i badawczej.

Jak widać, mimo tego, że autorowi udało się stworzyć kompleksowe rozwiązanie implementujące najważniejsze założenia programowania afektywnego, wciąż istnieje wiele obszarów, na których niniejszy projekt magisterski może być udoskonalony.

Bibliografia

[1] Olatz Lopez-Fernandez i in. „Self-reported dependence on mobile phones in young adults: A European cross-cultural empirical survey". *Journal of behavioral addictions*, 6(2), 2017, s. 168–177.

[2] Mark Alexa. „Amazon Alexa: 2017 User Guide+ 200 Ester Eggs", 2017.

[3] Robert Fisher, Asim Smailagic i George Sokos. „Monitoring Health Changes in Congestive Heart Failure Patients Using Wearables and Clinical Data". *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*. 2017, s. 1061–1064.

[4] Yi-Horng Lai i Fen-Fen Huang. „A Study on the Intention to Use the Wearable Device in Taiwan: A Case Study on Xiaomi Mi Band". *Proceedings of the Computational Methods in Systems and Software*. 2017, s. 283–292.

[5] Leora Trub i Baptiste Barbot. „The paradox of phone attachment: development and validation of the Young Adult Attachment to Phone Scale (YAPS)". *Computers in Human Behavior*, 64, 2016, s. 663–672.

[6] Manoj Shettar i in. „Facebook addiction and loneliness in the post-graduate students of a university in southern India". *International Journal of Social Psychiatry*, 63(4), 2017, s. 325–329.

[7] Daria J. Kuss. „OP-59: Social networking sites and social media addiction: Insights from current empirical research". *Journal of Behavioral Addictions*, 6(S1), 2017, s. 29–30.

[8] Przemysław Mastalerz. *Ekologiczne kłamstwa ekowojowników*. Wydawnictwo Chemiczne, 2012.

[9] Rosalind Wright Picard i in. „Affective computing", 1995.

[10] Luiz Moutinho i Mladen Sokele. „Innovative Research Methodologies in Management", 2018.

[11] Gartner Inc. „Hype Cycle for Education, 2017". *Hype Cycle Research Methodology*, 2017.

[12] Adam Węgrzecki. „O irracjonalności uczuć". *Zeszyty Naukowe Akademii Ekonomicznej w Krakowie*, 310, 1990, s. 5–17.

[13] Azadeh Rezvani i in. „Manager emotional intelligence and project success: The mediating role of job satisfaction and trust". *International Journal of Project Management*, 34(7), 2016, s. 1112–1122.

[14] James D. A. Parker i in. „Emotional intelligence and academic success: Examining the transition from high school to university". *Personality and individual differences*, 36(1), 2004, s. 163–172.

[15] Antoine Bechara, Hanna Damasio i Antonio R. Damasio. „Emotion, decision making and the orbitofrontal cortex". *Cerebral cortex*, 10(3), 2000, s. 295–307.

[16] Peter J. Lang. „Emotion and motivation: Toward consensus definitions and a common research purpose". *Emotion Review*, 2(3), 2010, s. 229–233.

[17] David R. Michael i Sandra L. Chen. „Serious games: Games that educate, train, and inform", 2005.

[18] Irene Kotsia, Ioannis Patras i Spiros Fotopoulos. *Affective gaming: Beyond using sensors*. 2012.

[19] Lennart Erik Nacke i in. „Biofeedback game design: using direct and indirect physiological control to enhance game interaction". *Proceedings of the SIGCHI conference on human factors in computing systems*. 2011, s. 103–112.

[20] Claire Dormann, Jennifer R. Whitson i Max Neuvians. „Once more with feeling: Game design patterns for learning in the affective domain". *Games and Culture*, 8(4), 2013, s. 215–237.

[21] Grzegorz J. Nalepa i in. „Affective Design Patterns in Computer Games. Scrollrunner Case Study". *Communication Papers of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017*. 2017, s. 345–352.

[22] David Capelo Chaves Caminha. „Development of Emotional Game Mechanics through the use of Biometric Sensors", 2017.

[23] Staffan Bjork i Jussi Holopainen. *Patterns in game design (game development series)*. Charles River Media, 2004.

[24] James W. Kalat. *Biological psychology*. Cengage Learning, 1980.

[25] Raúl Rojas i Ulf Hashagen. *The first computers: History and architectures*. MIT press, 2002.

[26] Martin H. Weik. *A third survey of domestic electronic digital computing systems*. Spraw. tech. Army Ballistic Researcg Lab Aberdeen Proving Ground MD, 1961.

[27] Nicholas Enticknap. „Computing's Golden Jubilee". *Resurrection: the computer conservation society*, (20), 1998.

[28] Jan Noyes. „The QWERTY keyboard: A review". *International Journal of Man-Machine Studies*, 18(3), 1983, s. 265–281.

[29] Paul E. Ceruzzi. *A history of modern computing*. MIT press, 2003.

[30] Joni Uitto i in. „Preventing malicious attacks by diversifying Linux shell commands." *SPLST*. 2015, s. 206–220.

[31] George M. Sheldrick. „A short history of SHELX". *Acta Crystallographica Section A: Foundations of Crystallography*, 64(1), 2008, s. 112–122.

[32] Jeremy Reimer. „A History of the GUI". *Ars Technica*, 5, 2005.

[33] Edward Nęcka, Jarosław Orzechowski i Błażej Szymura. *Psychologia poznawcza*. Academica Wydawnictwo SWPS, 2006.

[34] Apple Inc. „Apple Mac OS 1.0". *Apple Mac OS 1.0*, 2018.

[35] Pär-Anders Albinsson i Shumin Zhai. „High precision touch screen interaction". *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2003, s. 105–112.

[36] Neil Enns i Scott MacKenzie. „Touchpad-based remote control devices". *CHI 98 conference summary on Human factors in computing systems*. ACM. 1998, s. 229–230.

[37] Microsoft. „Microsoft researchers achieve new conversational speech recognition milestone - Microsoft Research". *Microsoft Store*, sierp. 2017.

[38] Hyunji Chung i Sangjin Lee. „Intelligent Virtual Assistant knows Your Life". *ArXiv*, 2018.

[39] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*. T. 25. MIT press, 1961.

[40] Google. „Google Duplex: An AI System for Accomplishing Real-World Tasks Over the Phone". *Google AI Blog*, maj 2018.

[41] John Markoff. *Machines of loving grace: The quest for common ground between humans and robots*. HarperCollins Publishers, 2016.

[42] Microsoft. „Microsoft HoloLens". *Mirosoft HoLolens for education*, 2018.

[43] Paul Milgram i Fumio Kishino. „A taxonomy of mixed reality visual displays". *IEICE Transactions on Information and Systems*, 77(12), 1994, s. 1321–1329.

[44] Sasha Azad i in. „Procedural Level Generation for Augmented Reality Games", 2015.

[45] Jurgen Ruesch i in. *Communication: The social matrix of psychiatry*. Routledge, 2017.

[46] S. G. Mason i in. „A comprehensive survey of brain interface technology designs". *Annals of biomedical engineering*, 35(2), 2007, s. 137–169.

[47] Rory Cellan-Jones. „Stephen Hawking warns artificial intelligence could end mankind". *BBC News*, 2, 2014, s. 2014.

[48] Amitai Etzioni i Oren Etzioni. „Should Artificial Intelligence Be Regulated?", 2017.

[49] William James. „What is an emotion?" *Mind*, 9(34), 1884, s. 188–205.

[50] Jan Strelau i Dariusz Doliński. „Psychologia". *Podręcznik akademicki*, 2(s 320), 2000, s. 444.

[51] Otniel E. Dror. „The Cannon–Bard thalamic theory of emotions: A brief genealogy and reappraisal". *Emotion Review*, 6(1), 2014, s. 13–20.

[52] Richard S. Lazarus. „Progress on a cognitive-motivational-relational theory of emotion." *American psychologist*, 46(8), 1991, s. 819.

[53] Robert B. Zajonc. „On the primacy of affect.", 1984.

[54] Joseph LeDoux. *The emotional brain: The mysterious underpinnings of emotional life*. Simon i Schuster, 1998.

[55] Jesse J. Prinz. *Gut reactions: A perceptual theory of emotion*. Oxford University Press, 2004.

[56] Paul Ekman. „Emocje ujawnione". *Odkryj, co ludzie chcą przed Tobą zataić, i dowiedz się czegoś więcej o sobie. Gliwice: Wydawnictwo Helion*, 2012.

[57] Robert Plutchik. *Emotion: A psychoevolutionary synthesis*. Harpercollins College Division, 1980.

[58] Anna Wierzbicka. *Cross-cultural pragmatics*. Walter de Gruyter Inc, 2003.

[59] Jianhua Tao i Tieniu Tan. „Affective computing: A review". *International Conference on Affective computing and intelligent interaction*. Springer. 2005, s. 981–995.

[60] Eva Hudlicka. „Affective computing for game design". *Proceedings of the 4th Intl. North American Conference on Intelligent Games and Simulation*. 2008, s. 5–12.

[61] Daniel Bersak i in. „Intelligent biofeedback using an immersive competitive environment". Paper at the Designing Ubiquitous Computing Games Workshop at UbiComp. 2001.

[62] Stacy C. Marsella i Jonathan Gratch. „EMA: A process model of appraisal dynamics". *Cognitive Systems Research*, 10(1), 2009, s. 70–90.

[63] Raúl Lara-Cabrera i David Camacho. „A taxonomy and state of the art revision on Affective Games". *Future Generation Computer Systems*, 2018.

[64] James A. Russell, Anna Weiss i Gerald A. Mendelsohn. „Affect grid: a single-item scale of pleasure and arousal." *Journal of personality and social psychology*, 57(3), 1989, s. 493.

[65] Pedro Gonçalo Ferreira Alves Nogueira. „Emotional State Regulation in Interactive Environments: A Psychophysiological Adaptive Approach for Affect-Inductive Experiences", 2016.

[66] Rosalind W. Picard. „Affective Computing for HCI." *HCI (1)*. 1999, s. 829–833.

[67] Artur Marchewka i in. „The Nencki Affective Picture System (NAPS): Introduction to a novel, standardized, wide-range, high-quality, realistic picture database". *Behavior research methods*, 46(2), 2014, s. 596–610.

[68] Ze-Jing Chuang i Chung-Hsien Wu. „Multi-modal emotion recognition from speech and text". *International Journal of Computational Linguistics & Chinese Language Processing, Volume 9, Number 2, August 2004: Special Issue on New Trends of Speech and Language Processing*, 9(2), 2004, s. 45–62.

[69] Paula M. Niedenthal i Marc B. Setterlund. „Emotion congruence in perception". *Personality and Social Psychology Bulletin*, 20(4), 1994, s. 401–411.

[70] 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". *User Modeling and User-Adapted Interaction*, 22(3), 2012, s. 255–279.

[71] Stefan Kopp i in. „Max-a multimodal assistant in virtual reality construction." *KI*, 17(4), 2003, s. 11.

[72] Luca Canini, Sergio Benini i Riccardo Leonardi. „Affective recommendation of movies based on selected connotative features". *IEEE Transactions on Circuits and Systems for Video Technology*, 23(4), 2013, s. 636–647.

[73] Sergio Benini, Luca Canini i Riccardo Leonardi. „A connotative space for supporting movie affective recommendation". *IEEE Transactions on Multimedia*, 13(6), 2011, s. 1356–1370.

[74] Lennart Nacke. „Affective ludology: Scientific measurement of user experience in interactive entertainment". Prac. dokt. Blekinge Institute of Technology, 2009.

[75] Charlene Jennett i in. „Measuring and defining the experience of immersion in games". *International journal of human-computer studies*, 66(9), 2008, s. 641–661.

[76] CD Projekt RED. „Wiedźmin 3: Dziki Gon". *Geralt rozmawiający z trollem*, 2018.

[77] Stephen Thompson. „The Open World Microorganism", 2018.

[78] Joel Lee. „7 Games You Can Mod to Add VR Support Right Now". *MakeUseOf*, mar. 2016.

[79] Boyan Bontchev. „Adaptation in affective video games: A literature review". *Cybernetics and Information Technologies*, 16(3), 2016, s. 3–34.

[80] Jonathan Sykes i Simon Brown. „Affective gaming: measuring emotion through the gamepad". *CHI'03 extended abstracts on Human factors in computing systems*. ACM. 2003, s. 732–733.

[81] S. P. Preejith i in. „Design, development and clinical validation of a wrist-based optical heart rate monitor". *Medical Measurements and Applications (MeMeA), 2016 IEEE International Symposium on*. IEEE. 2016, s. 1–6.

[82] Yoshiro Tajitsu. „Piezoelectret sensor made from an electro-spun fluoropolymer and its use in a wristband for detecting heart-beat signals". *IEEE Transactions on Dielectrics and Electrical Insulation*, 22(3), 2015, s. 1355–1359.

[83] Saewon Kye i in. „Multimodal data collection framework for mental stress monitoring". *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. ACM. 2017, s. 822–829.

[84] Krzysztof Kutt i in. „BandReader – A Mobile Application for Data Acquisition from Wearable Devices in Affective Computing Experiments". 2018.

[85] Eva Hudlicka. „Affective game engines: motivation and requirements". *Proceedings of the 4th international conference on foundations of digital games*. ACM. 2009, s. 299–306.

[86] Jonathan Tremblay i Clark Verbrugge. „Adaptive companions in FPS games." *FDG*, 13, 2013, s. 229–236.

[87] Sander Bakkes, Pieter Spronck i Jaap Van den Herik. „Rapid and reliable adaptation of video game AI". *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2), 2009, s. 93–104.

[88] Avinash Parnandi i Ricardo Gutierrez-Osuna. „A comparative study of game mechanics and control laws for an adaptive physiological game". *Journal on Multimodal User Interfaces*, 9(1), 2015, s. 31–42.

[89] Vanus Vachiratamporn i in. „An analysis of player affect transitions in survival horror games". *Journal on Multimodal User Interfaces*, 9(1), 2015, s. 43–54.

[90] Georgios N. Yannakakis i Julian Togelius. „Experience-driven procedural content generation". *IEEE Transactions on Affective Computing*, 2(3), 2011, s. 147–161.

[91] Robin Hunicke. „The case for dynamic difficulty adjustment in games". *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM. 2005, s. 429–433.

[92] Fabrizio Balducci, Costantino Grana i Rita Cucchiara. „Classification of affective data to evaluate the level design in a role-playing videogame". *Games and Virtual Worlds for Serious Applications (VS-Games), 2015 7th International Conference on*. IEEE. 2015, s. 1–8.

[93] Tim Marsh i in. „Automating the detection of breaks in continuous user experience with computer games". *CHI'05 Extended Abstracts on Human Factors in Computing Systems*. ACM. 2005, s. 1629–1632.

[94] Mary Shaw i David Garlan. *Software architecture: perspectives on an emerging discipline*. T. 1. Prentice Hall Englewood Cliffs, 1996.

[95] Ian Sommerville. *Inżynieria oprogramowania*. Wydawnictwa Naukowo-Techniczne, 2003.

[96] Bernard Suits. „The Grasshopper: Games, Life and Utopia 34", 2005.

[97] Kiel Gilleade, Alan Dix i Jen Allanson. „Affective videogames and modes of affective gaming: assist me, challenge me, emote me". *DiGRA 2005: Changing Views–Worlds in Play.* 2005.

[98] Ubisoft. „Assassin's Creed 3". *Connor używający Wzroku Orła*, 2018.

[99] Ian Millington. *Game physics engine development*. CRC Press, 2007.

[100] Rafał Rouba. „Realny sukces wirtualnego sportu-Intel Extreme Masters-Katowice". *Przegląd Nauk Ekonomicznych*, (27), 2017, s. 301–312.

[101] Rovio Mobile. „Angry Birds". *Fizyka pocisku jako główna mechanika*, 2018.

[102] Adrian Herwig i Philip Paar. „Game engines: tools for landscape visualization and planning". *Trends in GIS and virtualization in environmental planning and design*, 161, 2002, s. 172.

[103] Faizi Noor Ahmad. „An Overview Study of Game Engines", 2013.

[104] Crytek. „CryENGINE V Manual". *Documentation*, 2018.

[105] Crytek. „Crysis 3". *Fotorealistyczna scena*, 2018.

[106] Paul E. Dickson i in. „An Experience-based Comparison of Unity and Unreal for a Stand-alone 3D Game Development Course". *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ACM. 2017, s. 70–75.

[107] Game Editor. „Welcome to Game Editor". *Documentation*, 2018.

[108] Edward Byrne. *Game level design*. T. 6. Charles River Media Boston, 2005.

[109] Joanne O'Mara i John Richards. „A blank slate of potential: using GameMaker to create computer games". *Digital Games: literacy in action*, 2012, s. 57–64.

[110] GameMaker. *Welcome to GameMaker*. 2018.

[111] Godot. „Godot Docs". *Godot Engine latest documentation*, 2018.

[112] Lars Bergqvist. „Godot Tutorial". *Tworzenie poziomu w oparciu o silnik Godot*, 2018.

[113] Katarzyna Grzesiak-Kopeć, Leszek Nowak i Maciej Ogorzałek. „3D Integrated Circuits Layout Optimization Game". *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2017, s. 444–453.

[114] Unity Technologies. „Unity User Manual (2018.1)". *Unity - Scripting API: Physics.Raycast*, 2018.

[115] Unity. „Unity - środowisko programistyczne". *Interfejs programistyczny*, 2018.

[116] „Unreal Engine 4 Documentation". *Introduction to Blueprints*, 2018.

[117] Irrational games. „Bioshock Infinite". *Walka z pozycji pierwszosobowej*, 2018.

[118] Christopher Alexander. *A pattern language: towns, buildings, construction*. Oxford University Press, 1977.

[119] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[120] N. Ravaja i in. „Emotional response patterns and sense of presence during video games: Potential criterion variables for game design". *Proceedings of the third Nordic conference on Human-computer interaction*. ACM. 2004, s. 339–347.

[121] N. Ravaja i in. „The psychophysiology of video gaming: Phasic emotional responses to game events [CD-ROM]". *Proceedings of DiGRA 2005 Conference*. 2005.

[122] Kotaku. „This War of Mine". *Depresyjna odmiana The Sims*, 2018.

[123] Iwona Grabska-Gradzińska i Jan K. Argasiński. „Patterns in Video Games Analysis–Application of Eye-Tracker and Electrodermal Activity (EDA) Sensor". *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2018, s. 619–629.

[124] Edwin G. Boring. „A history of introspection." *Psychological bulletin*, 50(3), 1953, s. 169.

[125] Power of Positivy. „Mapy cieplne". *Rozróżnianie emocji podstawowych*, 2018.

[126] Keith W. Jacobs i Frank E. Hustmyer Jr. „Effects of four psychological primary colors on GSR, heart rate and respiration rate". *Perceptual and motor skills*, 38(3), 1974, s. 763–766.

[127] Christine L. Lisetti i Diane J. Schiano. „Automatic facial expression interpretation: Where human-computer interaction, artificial intelligence and cognitive science intersect". *Pragmatics & cognition*, 8(1), 2000, s. 185–235.

[128] Lennart E. Nacke i Regan L. Mandryk. „Designing affective games with physiological input". *Workshop on Multiuser and Social Biosignal Adaptive Games and Playful Applications in Fun and Games Conference (BioS-Play)*. 2010.

[129] Alexandre Tuch i in. „The role of visual complexity in affective reactions to webpages: Subjective, eye movement, and cardiovascular responses". *IEEE transactions on affective computing*, 2(4), 2011, s. 230–236.

[130] BITalino. „Respiracja". *Opaska do pomiaru respiracji*, 2018.

[131] Shigeru Sakurazawa, Naofumi Yoshida i Nagisa Munekata. „Entertainment feature of a game using skin conductance response". *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM. 2004, s. 181–186.

[132] Héctor Perez Martinez, Maurizio Garbarino i Georgios N. Yannakakis. „Generic physiological features as predictors of player experience". *International Conference on Affective Computing and Intelligent Interaction*. Springer. 2011, s. 267–276.

[133] Honors Fellows. „Przewodnictwo skórne". *Elektrody umożliwiające pomiar reakcji elektrodermalnej*, 2018.

[134] My Med World. „Elektrokardiogram". *Załamki, odcinki i zespoły*, 2018.

[135] Bio Radio. „Elektromiogram". *Sygnał z elektromiografu*, 2018.

[136] Gonçalo Pereira i in. „A generic emotional contagion computational model". *International Conference on Affective Computing and Intelligent Interaction*. Springer. 2011, s. 256–266.

[137] W. M. Van den Hoogen i in. „Toward real-time behavioral indicators of player experiences: Pressure patterns and postural responses". *Proceedings of Measuring Behaviour 2008*, 2008, s. 100–101.

[138] Kiyoung Yang, Tim Marsh i Cyrus Shahabi. „Continuous archival and analysis of user data in virtual and immersive game environments". *Proceedings of the 2nd ACM workshop on Continuous archival and retrieval of personal experiences*. ACM. 2005, s. 13–22.

[139] Moataz El Ayadi, Mohamed S. Kamel i Fakhri Karray. „Survey on speech emotion recognition: Features, classification schemes, and databases". *Pattern Recognition*, 44(3), 2011, s. 572–587.

[140] Stylianos Asteriadis i in. „Towards player's affective and behavioral visual cues as drives to game adaptation". *LREC Workshop on Multimodal Corpora for Machine Learning*. Citeseer. 2012, s. 6–9.

[141] Alessandro Canossa. *Play-Persona: Modeling Player Behavior in Computer Games*. Danmarks Designskole, 2009.

[142] Fine Touch Marketing. „Mapa cieplna". *Kliknięcia na stronie internetowej*, 2018.

[143] NeXus. „Nexus-10 User Manual". *NeXus-10 Documentation*, 2018.

[144] Neurobit. „Neurobit Optima User Manual". *Neurobit Documentation*, 2018.

[145] Fatema El-Amrawy i Mohamed Ismail Nounou. „Are currently available wearable devices for activity tracking and heart rate monitoring accurate, precise, and medically beneficial?" *Healthcare informatics research*, 21(4), 2015, s. 315–320.

[146] e-Health. „e-Health Sensor Platform Documentation". *e-Health Sensor Platform*, 2018.

[147] BITalino. „BITalino (r)evolution kit Documentation". *BITalino - Biomedical Equipment a Low-Cost Toolkit*, 2018.

[148] Regan L. Mandryk i M. Stella Atkins. „A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies". *International journal of human-computer studies*, 65(4), 2007, s. 329–347.

[149] Joseph Onderi Orero i in. „Assessing gameplay emotions from physiological signals: A fuzzy decision trees based model". *International Conference on Kansei Engineering and Emotion Research 2010*. 2010, s. 1684–1693.

[150] Longfei Li i in. „Hybrid Deep Neural Network–Hidden Markov Model (DNN-HMM) Based Speech Emotion Recognition". *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*. IEEE. 2013, s. 312–317.

[151] Mohammad H. Sedaaghi, Constantine Kotropoulos i Dimitrios Ververidis. „Using adaptive genetic algorithms to improve speech emotion recognition". *Multimedia Signal Processing, 2007. MMSP 2007. IEEE 9th Workshop on*. IEEE. 2007, s. 461–464.

[152] Chi-Chun Lee i in. „Emotion recognition using a hierarchical binary decision tree approach". *Speech Communication*, 53(9-10), 2011, s. 1162–1171.

[153] Oh-Wook Kwon i in. „Emotion recognition by speech signals". *Eighth European Conference on Speech Communication and Technology*. 2003.

[154] George Kennedy i in. „Technology Solutions to Combat Online Harassment". *Proceedings of the First Workshop on Abusive Language Online*. 2017, s. 73–77.

[155] Isreael 21c. „Starsi ludzie grający w grę wideo". *Radość na twarzach ludzi grających w grę*, 2018.

[156] Andreas Haag i in. „Emotion recognition using bio-sensors: First steps towards an automatic system". *Tutorial and research workshop on affective dialogue systems*. Springer. 2004, s. 36–48.

[157] Boise State University. „Rozpoznawanie emocji za pomocą cech twarzy". *Twarz a emocje*, 2018.

[158] Unity3D. „Survival Shooter Tutorial". *YouTube*, 2014.

[159] Unity. „Survival Shooter Tutorial". *Unity Assets Store*, 2017.

[160] BITalino. „BITalino (r)evolution kit APIs". *BITalino - Biomedical Equipment APIs*, 2018.

[161] Flipe Rodrigues. „BITalino - Unity Integration". *GitHub*, 2015.

[162] Filipe Rodrigues. „Unity-Bitalino Tutorial". *YouTube*, 2015.

[163] Christtopher Watford. „Understanding ECG Filtering". *EMS 12 Lead*, 2014.

[164] Joe Pololu. „Pololu". *Robotics and Electronics*, 2018.

[165] Marieke van Dooren i Joris H. Janssen. „Emotional sweating across the body: Comparing 16 different skin conductance measurement locations". *Physiology & behavior*, 106(2), 2012, s. 298–304.

Dodatki

Dodatek A

Fragment skryptu PlayerShooting.cs:

```
using System;
using System.Collections;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;


/*
    Odpowiada za możliwość ataku gracza.
*/
public class PlayerShooting : MonoBehaviour
{
    /*
        Parametry ataku gracza.
    */
    public int damagePerShot = 20;
    public int damagePerShotMin = 10;

    public float timeBetweenBullets = 0.2f;
    public float timeBetweenBulletsMax = 0.8f;

    public float range = 100f;
    public float rangeMin = 40f;

    public float ultraPower = 100;

    /*
        Regeneracja zdolności gracza.
    */
```

```
float regenerationTimer = 0f;

float ultraPowerRegeneration = 5f;
float rangeRenegeration = 10f;
float timeBetweenBulletsRegeneration = 0.1f;
int damagePerShotRegeneration = 1;

/*
    Zdolność SuperMocy i jej reprezentacja wizualna.
*/
public GameObject enemy;
public GameObject[] enemies;
public Slider powerSlider;
public Image superpowerImage;

bool superpowered = false;

float timeAlpha = 0f;
float duration = 10000000;
float smoothness = 0.02f;


/*
    Umożliwia aktywację SuperMocy.
    Parametry: timeSuper - czas użycia SuperMocy,
               timeMax - czas maksymalny trwania SuperMocy,
               was - czy stwierdzono użycie SuperMocy?
*/
void UseSuperPower(float timeSuper, float timeMax, bool was)
{
    if((timeSuper > 0 && was)  && ultraPower > 0)
    {
        LogManager.logManager.AddEvent(Time.time,
            "Player;SuperPower;Time;" + timeSuper);

        float ultraPowerused = ultraPower * (Math.Min(timeSuper,
            timeMax) / timeMax);
        ultraPower -= ultraPowerused;
        powerSlider.value = ultraPower;
        LogManager.logManager.AddEvent(Time.time,
            "Slider;SuperPower;DecreaseTo;" + ultraPower);

        timeBetweenBullets += timeBetweenBulletsRegeneration;
```

```
if(timeBetweenBullets > timeBetweenBulletsMax)
{
    timeBetweenBullets = timeBetweenBulletsMax;
}
LogManager.logManager.AddEvent(Time.time,
    "Player;TimeBetweenTwoBullets;IncreaseTo;" +
    timeBetweenBullets);

damagePerShot -= damagePerShotRegeneration;
if(damagePerShot < damagePerShotMin)
{
    damagePerShot = damagePerShotMin;
}
LogManager.logManager.AddEvent(Time.time,
    "Player;DamagePerShot;DecreaseTo;" + damagePerShot);

range -= rangeRenegeration;
if(range < rangeMin)
{
    range = rangeMin;
}
LogManager.logManager.AddEvent(Time.time,
    "Player;Range;DecreaseTo;" + range);

timeAlpha = timeSuper;
regenerationTimer = 0;

timeSuper = 0;
was = false;

enemies = GameObject.FindGameObjectsWithTag("Enemy");

int killed = UnityEngine.Random.Range (0, Math.Min(2 *
    (int)ultraPowerused, enemies.Length));

LogManager.logManager.AddEvent(Time.time,
    "Player;SuperPower;Kill;" + killed);

for(int i = 0; i < killed; i++)
{
    enemy = enemies[i];
    EnemyHealth enemyHealth = enemy.GetComponent <EnemyHealth>
        ();
```

```
                enemyHealth.TakeDamageSuper(enemyHealth.currentHealth);
            }

            superpowered = true;

        }

        if(superpowered)
        {
            float alpha = Math.Min(timeAlpha, timeMax)/timeMax;
            LogManager.logManager.AddEvent(Time.time,
                "Player;SuperPower;Alpha;" + alpha);
            Color flashColour = new Color(1f, 0.588235f, 0f, alpha);
            superpowerImage.color = flashColour;
            StartCoroutine(LerpColor());
        }
        superpowered = false;
    }

    /*
        Umożliwia aktywację wizualnych efektów SuperMocy.
        Zwracanie: iterator, ze względu na opóźnienia czasowe.
    */
    IEnumerator LerpColor()
    {
        float progress = 0;
        float increment = smoothness / duration;
        while(progress < 1)
        {
            superpowerImage.color = Color.Lerp(superpowerImage.color,
                Color.clear, progress);
            progress += increment;
            yield return new WaitForSeconds(smoothness);
        }
}
```

Fragment skryptu CamMouseLook.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

/*
    Odpowiada za zarządzanie ruchem myszy w widoku fps.
```

```
*/
public class CamMouseLook : MonoBehaviour
{
    /*
        Ograniczenia i fizyka ruchu.
        Umożliwienie edycji w środowisku Unity.
    */
    public float speed = 5f;
    public float turn = 3f;
    public float limit_up = 30;
    public float limit_down = 30;
    public float limit_three = 60;

    /*
        Fizyka ruchu.
    */
    GameObject parentObj;
    Rigidbody otherRb;
    Vector3 angles;


    /*
        Nadanie wartości początkowych.
    */
    void Awake ()
    {
        Cursor.lockState = CursorLockMode.Locked;
        parentObj = this.transform.parent.gameObject;
        otherRb = parentObj.GetComponent <Rigidbody> ();
        float xstart = Input.mousePosition.x;
        float ystart = Input.mousePosition.y;
    }


    /*
        Aktualizacja pozycji kamery w oparciu
        o dane pochodzące z sensorów myszy.
    */
    void FixedUpdate ()
    {
        float xturn = Input.GetAxis("Mouse X");
        float yturn = Input.GetAxis ("Mouse Y");
```

```
        parentObj.transform.Rotate (0f, xturn * turn, 0f);
        if (angles.x <= limit_up && angles.x >= limit_down)
        {
            transform.Rotate (-yturn * turn, 0f, 0f);
        }
        else if(angles.x < limit_down)
        {
            angles.x = limit_down + 1;
            transform.localEulerAngles = angles;
        }

        else if (angles.x < limit_three)
        {
            angles.x = limit_up - 1;
            transform.localEulerAngles= angles;
        }

        else
        {
            angles.x = limit_down + 1;
            transform.localEulerAngles = angles;
        }

        angles = transform.localEulerAngles;

    }
}
```

Fragment skryptu ImportantAlertManager.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


/*
        Odpowiada za zarządzanie obiektem wyświetlającym
        ostrzeżenia i informacje na środku ekranu.
*/
public class ImportantAlertManager : MonoBehaviour
{
        /*
                Realizacja wzorca Singleton.
```

```
*/
public static ImportantAlertManager importantAlertManager;


/*
        Uruchomienie aktywacji wzorca Singleton.
*/
void Awake ()
{
        MakeThisTheOnlyDontDestroyManager();
}


/*
        Aktywacja wzorca Singleton.
*/
void MakeThisTheOnlyDontDestroyManager()
{

        if(importantAlertManager == null)
        {
                DontDestroyOnLoad(gameObject);
                importantAlertManager = this;
        }

        else
        {
                if(importantAlertManager != this)
                {
                        Destroy (gameObject);
                }
        }
}


/*
        Umożliwia pokazanie alertu na ekranie
        oraz jego stopniowe zanikanie.
        Parametry: time - czas widoczności alertu
                        input - treść pokazywanego alertu
        Zwracanie: iterator, ze względu na opóźnienia czasowe.
*/
public IEnumerator ShowAlertAndLerp(float time, string input)
```

```
        {
                Text txt = gameObject.GetComponent<Text>();
                txt.enabled = true;

                txt.text = input;

                Color startingColor = Color.green;
                txt.color = startingColor;

                float inversedTime = 1 / time;
                for(float step = 0.0f; step < 1.0f; step += Time.deltaTime
                    * inversedTime)
                {
                        txt.color = Color.Lerp(startingColor, Color.clear,
                            step);
                        yield return null;
                }

                txt.enabled = false;
        }


        /*
                Umożliwia pokazanie alertu na ekranie,
                bez stopniowego zanikania.
                Parametry: time - czas widoczności alertu
                                input - treść pokazywanego alertu
                Zwracanie: iterator, ze względu na opóźnienia czasowe.
        */
        public IEnumerator ShowAlert(float time, string input)
        {
                Text txt = gameObject.GetComponent<Text>();
                txt.enabled = true;

                txt.text = input;

                yield return new WaitForSeconds(time);

                txt.enabled = false;
        }
}
```

Fragment skryptu ManagerScoreBoard.cs:

```
using System;
using System.IO;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using UnityEditor;

/*
    Odpowiada za zarządzanie tablicą wyników.
*/
public class ManageScoreBoard : MonoBehaviour
{
        /*
                Referencja do obiektu wyświetlającego tekst.
        */
        public Text textBestPlayers;

        /*
                Zapis wyników do pliku
        */
        string pathScores = "Assets\\Datalogs\\Scores.csv";
        Dictionary<int, List<string>> bestPlayers = new Dictionary<int,
            List<string>> ();


        /*
                Odczytuje najlepsze wyniki z pliku.
        */
        void ReadBestPlayers()
        {
                if (File.Exists(pathScores))
                {
                        StreamReader reader = new StreamReader(pathScores);
                        string line;
                        while((line = reader.ReadLine()) != null)
                        {
                                string[] elements = line.Split(';');
```

```
                        int score = 0;
                        Int32.TryParse(elements[0], out score);

                        List<string> names = new List<string> ();
                        for (int i = 1; i < (elements.Length - 1);
                            i++)
                        {
                                names.Add(elements[i]);
                        }

                        bestPlayers.Add(score, names);
                }

                reader.Close();
        }
}


/*
        Dodaje wynik obecnego gracza do listy wyników.
*/
void AddNewPlayer()
{
        bool bitalinoUse =
            BitalinoController.bitalinoController.bitalinoUse;
        string use = bitalinoUse ? "1" : "0";
        string currentName = UserManager.userManager.GetUserName()
            + " (" + use + ")";
        int currentScore =
            UserManager.userManager.GetCumulatedScore();

        if (!bestPlayers.ContainsKey(currentScore))
        {
                List<string> players = new List<string> ();
                players.Add(currentName);
                bestPlayers.Add(currentScore, players);
        }

        else
        {
                List<string> players = new List<string> ();
                players = bestPlayers[currentScore];
```

```
                bestPlayers.Remove(currentScore);
                players.Add(currentName);
                bestPlayers.Add(currentScore, players);
        }
}


/*
        Wypisuje najlepszych graczy wraz z ich wynikami
        na ekranie monitora.
*/
void PrintBestPlayers()
{
        string bests = "";
        int i = 0;

        List<int> keyList = bestPlayers.Keys.ToList();
        keyList.Sort();
        keyList.Reverse();
        foreach (int key in keyList)
        {
                foreach (string el in bestPlayers[key])
                {
                        i++;
                        bests += String.Format("{0}. {1} {2}\n" ,
                            i, el, key);
                }
        }

        textBestPlayers.text = bests;
}


/*
        Zapisuje najlepsze wyniki do pliku.
*/
void WriteBestPlayers()
{
        if(File.Exists(pathScores))
        {
                File.Delete(pathScores);
        }
```

```
                StreamWriter writer = new StreamWriter(pathScores);

                List<int> keyList = bestPlayers.Keys.ToList();
                foreach (int key in keyList)
                {
                        string bests = "";
                        foreach (string el in bestPlayers[key])
                        {
                                bests += String.Format("{0};", el);
                        }

                        string line = String.Format("{0};", key);
                        line += bests;

                        writer.WriteLine(line);
                }

                writer.Close();
        }
}
```

Fragment skryptu BitalinoController.cs:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.SceneManagement;
using MathNet.Filtering;
using System.IO;


/*
    Odpowiada za zarządzanie obiektem reprezentującym
    BITalino (r)evolution kit w świecie gry.
*/
public class BitalinoController : MonoBehaviour
{
    /*
        Referencja do innych skryptów biorących udział w zarządzaniu
        BITalino (r)evolution kit.
    */
    public BitalinoManager bitalinoManager;
```

```
public BitalinoReader bitalinoReader;

/*
    Umożliwienie podglądu i zarządzania parametrami połączenia
    z platformą BITalino (r)evolution kit.
*/
public int ECGChannel = 0;
public int EMGChannel = 1;
public int EDAChannel = 2;

[Range(0f, 10f)]
public float cutoffLowpass = 5f;
[Range(0f, 1f)]
public float cutoffHighpass = 0.67f;

[Range(0,2000)]
public int calibrationSamples = 1000;
public float calibrationPreTime = 10;

public float EMGCalibrated = 0;
public float EDACalibrated = 0;
public float HRCalibrated = 0;

public float EMGMin = 0f;
public float EDAMin = 0f;
public float HRMin = 0f;
public float EMGMax = 0f;
public float EDAMax = 0f;
public float HRMax = 0f;

[HideInInspector]
public bool bitalinoUse = false;

[HideInInspector]
public int bufferSize;

/*
    Zarządzanie kalibracją.
*/
bool calibrated;

float ECGMax;
float ECGPeakThreshold = 0f;
```

```
float ECGPeakTimer = 0f;
bool ECGPeakFound = false;
LinkedList<float> HRs = new LinkedList<float>();
float HRAv;

/*
    Progi filtracji sygnałów.
*/
float cutoffEMGHigher = 3f;
float cutoffEDALower = 0.2f;
float cutoffEDAHigher = 0.2f;
float cutoffHRLower = 15f;
float cutoffHRHigher = 15f;


/*
    Rozpoczyna połączenie z platformą BITalino (r)evolution kit.
    Uruchamia i zarządza kalibracją sygnałów HR, EMG, EDA.
    Zwracanie: iterator, ze względu na opóźnienia czasowe.
*/
public IEnumerator StartReading()
{
    bitalinoReader.enabled = true;
    bufferSize = bitalinoReader.BufferSize;

    bitalinoUse = true;

    while (!bitalinoReader.asStart)
    {
        yield return new WaitForSeconds(0.2f);
    }
    yield return new WaitForSeconds(calibrationPreTime);

    List<BITalinoFrame> calibrationFrames = new List<BITalinoFrame>();

    int EMGCalibrationSamples = 0;
    int EDAcalibrationSamples = 0;
    int ECGcalibrationSamples = 0;

    float HRAvSum = 0;

    for (int i = 0; i < calibrationSamples ; i++)
    {
```

```
            BITalinoFrame[] buffer = bitalinoReader.getBuffer();

            if (i < calibrationSamples / 4)
            {
                yield return new WaitForSeconds(0.01f);
            }

            float current = EMGCalculateAverage(buffer);
            if (current < 3f)
            {
                EMGCalibrated += current;
                EMGCalibrationSamples++;
            }

            EDACalibrated += EDACalculateAverage(buffer);
            EDAcalibrationSamples ++;

            ECGCalculateHR(buffer.ToArray());
                    HRAvSum += HRAverage;
            ECGcalibrationSamples++;

            yield return new WaitForSeconds(0.01f);
        }

        EMGCalibrated = EMGCalibrated / EMGCalibrationSamples;
        EDACalibrated = EDACalibrated / EDAcalibrationSamples;
        HRCalibrated = HRAvSum / ECGcalibrationSamples;

        EMGMax = EMGCalibrated;
        EMGMin = EMGCalibrated;
        EDAMin = EDACalibrated;
        EDAMax = EDACalibrated;
        HRMax = HRCalibrated;
        HRMin = HRCalibrated;

        calibrated = true;
    }


    /*
        Oblicza średnią arytmetyczną ze wszystkich
        zgromadzonych próbek EMG.
        Parametr: buffer - bufor danych z BITalino (r)evolution kit.
```

```
*/
float EMGCalculateAverage(BITalinoFrame[] buffer)
{
    return buffer.Select(x =>
        Mathf.Abs((float)x.GetAnalogValue(EMGChannel))).Average();
}


/*
    Oblicza średnią arytmetyczną ze wszystkich
    zgromadzonych próbek EDA.
    Parametr: buffer - bufor danych z BITalino (r)evolution kit.
*/
float EDACalculateAverage(BITalinoFrame[] buffer)
{
    return buffer.Select(x =>
        Mathf.Abs((float)x.GetAnalogValue(EDAChannel))).Average();
}


/*
    Przelicza sygnał EKG na HR.
    Parametr: buffer - bufor danych z BITalino (r)evolution kit.
    Zwracanie: tablica wartości HR.
*/
double [] ECGCalculateHR (BITalinoFrame[] buffer)
    {
    OnlineFilter filter =
        OnlineFilter.CreateBandpass(ImpulseResponse.Infinite,
        bitalinoManager.SamplingFrequency, cutoffLowpass,
        cutoffHighpass);
    double[] filteredSamples = filter.ProcessSamples(buffer.Select(x
        => x.GetAnalogValue(ECGChannel)).ToArray());
    double[] valuesHR = new double[buffer.Length];

    float sample = 0f;
    float samplingFrequency = bitalinoManager.SamplingFrequency;

    for (int i = 0; i < buffer.Length; i++)
    {
        sample = (float)filteredSamples[i];

        if (sample > ECGMax)
```

```
            {
                ECGMax = sample;
            }

            ECGPeakThreshold = ECGMax * 0.5f;

            if (sample < ECGPeakThreshold && ECGPeakTimer > 1.5f)
            {
                ECGMax -= 0.01f;
            }

            if (sample > ECGPeakThreshold * 1.1f && !ECGPeakFound)
            {
                float HR = 60 / ECGPeakTimer;

                if (HR > 40 && HR < 130)
                {
                    if (HRs.Count >= 100)
                    {
                        HRs.RemoveLast();
                    }

                    HRs.AddFirst(HR);
                    HRAv = HRs.Average();
                }

                ECGPeakTimer = 0;
                ECGPeakFound = true;
            }

            if (sample < ECGPeakThreshold && ECGPeakFound)
            {
                ECGPeakFound = false;
            }

            ECGPeakTimer += 1 / samplingFrequency;

            valuesHR[i] = HRs.Count == 0 ? 0 : HRs.First.Value;
        }

        return valuesHR;
        }
}
```

Dodatek B

Instrukcja dla osoby badanej.

Dziękujemy, że zgodziłeś/-łaś się wziąć udział w naszym badaniu. Twoim zadaniem będzie gra w dwie wersje prostej gry komputerowej przez określony czas. Nie martw się, całość badania nie powinna przekroczyć 40 minut. W pierwszej części będziesz grać ze założonym sprzętem, a w drugiej bez niego.

Wszystkie pozostałe informacje zostaną przedstawione podczas rozgrywki. Dowiesz się jak się poruszać, co symbolizują elementy interfejsu oraz co jest Twoim celem.

Najważniejsze jest to, żebyś był/-a świadoma tego, że musisz przyswoić polecenia pojawiające się na ekranie i przestrzegać wszystkich poleceń eksperymentatora.

Na koniec poprosimy Cię o wypełnienie krótkiej ankiety, która pomoże nam w osiągnięciu naszych celów badawczych.

W razie wątpliwości, zadaj teraz swoje pytania osobie, która nadzoruje eksperyment. Powiadom ją też o zakończeniu czytania tego tekstu. Pokaże Ci, w jaki sposób założyć sprzęt oraz uruchomi odpowiednie wersje gry.

Pamiętaj, że możesz przerwać badanie w każdej chwili. Jeśli jednak potrzebujesz jedynie chwili odpoczynku, prosimy, abyś poczekał/-a z tym do zdjęcia urządzeń pomiarowych.

Po zakończeniu badania, jeśli będziesz chciał/-a udzielimy Ci wszelkich informacji na temat wszystkich elementów badania.

Dodatek C

Kwestionariusz po zakończeniu eksperymentu.

Pierwsza wersja gry to gra z pętlą afektywną, czyli **z** oprzyrządowaniem pomiarowym.
Druga wersja gry to gra bez pętli afektywnej, czyli **bez** oprzyrządowania pomiarowego.

1. **Oceń poziom trudności pierwszej wersji gry.**

   (a) Łatwy.

(b) Pomiędzy łatwym a średnim.

(c) Średni.

(d) Pomiędzy średnim a trudnym.

(e) Trudny.

2. Oceń poziom trudności drugiej wersji gry.

(a) Łatwy.

(b) Pomiędzy łatwym a średnim.

(c) Średni.

(d) Pomiędzy średnim a trudnym.

(e) Trudny.

3. Czy sprzęt pomiarowy przeszkadzał Ci podczas gry?

(a) Nie.

(b) Raczej nie.

(c) Trudno powiedzieć.

(d) Raczej tak.

(e) Tak.

4. Zaznacz wszystkie elementy gry, które zauważyłeś.

(a) Przedstawiana historia.

(b) Ukrycie informacji o dokładnej liczbie punktów.

(c) Ukrycie informacji o dokładnej wartości poziomu życia.

(d) Rozwój bohatera - pojawienie się *SuperMocy*.

(e) Pojawianie się ostrzeżeń/informacji na ekranie.

(f) Zmiana widoku z oddali na widok pierwszosobowy.

(g) Wrogowie i walka.

(h) Przeszkody i ich omijanie.

(i) Pojawienie się nowych rodzajów wrogów.

(j) Losowe miejsca pojawiania się przeciwników.

(k) Aktywacja dodatkowych miejsc pojawiania się przeciwników.

(l) Zmiana szybkości poruszania się przeciwników.

(m) Możliwość sprawdzenia i porównania swojego wyniku z innymi graczami.

5. Wybierz wszystkie elementy gry, które redukowały Twoje znudzenie i/lub zrelaksowanie.

(a) Przedstawiana historia.

(b) Ukrycie informacji o dokładnej liczbie punktów.

(c) Ukrycie informacji o dokładnej wartości poziomu życia.

(d) Rozwój bohatera - pojawienie się *SuperMocy*.

(e) Pojawianie się ostrzeżeń/informacji na ekranie.

(f) Zmiana widoku z oddali na widok pierwszosobowy.

(g) Wrogowie i walka.

(h) Przeszkody i ich omijanie.

(i) Pojawienie się nowych rodzajów wrogów.

(j) Losowe miejsca pojawiania się przeciwników.

(k) Aktywacja dodatkowych miejsc pojawiania się przeciwników.

(l) Zmiana szybkości poruszania się przeciwników.

(m) Możliwość sprawdzenia i porównania swojego wyniku z innymi graczami.

6. Zaznacz wszystkie sposoby działania gry, których doświadczyłeś.

(a) Użycie *SuperMocy* za pomocą napięcia bicepsa w pierwszej wersji gry.

(b) Użycie *SuperMocy* za pomocą naciśnięcia klawisza *K* w drugiej wersji gry.

(c) Aktywacja dodatkowych miejsc pojawiania się przeciwników w pierwszej wersji gry.

(d) Aktywacja dodatkowych miejsc pojawiania się przeciwników w drugiej wersji gry.

(e) Dostosowanie szybkości poruszania się przeciwników w pierwszej wersji gry.

(f) Zmiana szybkości poruszania się przeciwników w drugiej wersji gry.

(g) Dostosowanie losowości miejsc pojawiania się przeciwników w pierwszej wersji gry.

(h) Całkowicie losowe miejsca pojawiania się przeciwników w drugiej wersji gry.

7. Wybierz dwa ulubione sposoby działania gry.

(a) Użycie *SuperMocy* za pomocą napięcia bicepsa w pierwszej wersji gry.

(b) Użycie *SuperMocy* za pomocą naciśnięcia klawisza *K* w drugiej wersji gry.

(c) Aktywacja dodatkowych miejsc pojawiania się przeciwników w pierwszej wersji gry.

(d) Aktywacja dodatkowych miejsc pojawiania się przeciwników w drugiej wersji gry.

(e) Dostosowanie szybkości poruszania się przeciwników w pierwszej wersji gry.

(f) Zmiana szybkości poruszania się przeciwników w drugiej wersji gry.

(g) Dostosowanie losowości miejsc pojawiania się przeciwników w pierwszej wersji gry.

(h) Całkowicie losowe miejsca pojawiania się przeciwników w drugiej wersji gry.

8. Który sposób działania gry sprawił, że bardziej się w nią *wczułeś?*

(a) Użycie *SuperMocy* za pomocą napięcia bicepsa.

(b) Użycie *SuperMocy* za pomocą naciśnięcia klawisza *K*.

(c) Bez różnicy.

9. Który sposób działania gry sprawił, że bardziej się w nią *wczułeś?*

(a) Aktywacja dodatkowych miejsc pojawiania się przeciwników po określonym czasie.

(b) Aktywacja dodatkowych miejsc pojawiania się przeciwników po wykryciu Twojego znudzenia.

(c) Bez różnicy.

10. Który sposób działania gry sprawił, że bardziej się w nią *wczułeś?*

(a) Zwiększenie szybkości poruszania się przeciwników po czasie.

(b) Dostosowanie szybkości poruszania się przeciwników pod wpływem zmianny tętna.

(c) Bez różnicy.

11. Który sposób działania gry sprawił, że bardziej się w nią *wczułeś?*

(a) Dostosowanie losowości pojawiania się przeciwników w zależności od stanu emocjonalnego.

(b) Losowe pojawianie się przeciwników.

(c) Bez różnicy.

12. Który sposób działania gry sprawił, że poziom gry był bardziej dostosowany do Twoich możliwości?

(a) Aktywacja dodatkowych miejsc pojawiania się przeciwników po określonym czasie.

(b) Aktywacja dodatkowych miejsc pojawiania się przeciwników po wykryciu Twojego znudzenia.

(c) Bez różnicy.

13. Który sposób działania gry sprawił, że poziom gry był bardziej dostosowany do Twoich możliwości?

(a) Zwiększenie szybkości poruszania się przeciwników wraz z upływem czasu.

(b) Dostosowanie szybkości poruszania się przeciwników pod wpływem zmianny tętna.

(c) Bez różnicy.

14. Który sposób działania gry sprawił, że poziom gry był bardziej dostosowany do Twoich możliwości?

(a) Dostosowanie losowości miejsc pojawiania się przeciwników w zależności od stanu emocjonalnego.

(b) Całkowicie losowe miejsca pojawiania się przeciwników.

(c) Bez różnicy.

15. Który sposób działania gry sprawił, że czułeś większą kontrolę?

(a) Użycie *SuperMocy* za pomocą napięcia bicepsa.

(b) Użycie *SuperMocy* za pomocą naciśnięcia klawisza *K*.

(c) Bez różnicy.

16. Miejsce na Twoje uwagi:

Dodatek D

Wykaz elementów znajdujących się na dołączonej płycie CD.

1. Tekst niniejszej pracy magisterskiej w formacie PDF.