



WYDZIAŁ ELEKTRONIKI,
TELEKOMUNIKACJI
I INFORMATYKI

Imię i nazwisko studenta: Karol Lange

Nr albumu: 175921

Poziom kształcenia: Studia drugiego stopnia

Forma studiów: niestacjonarne

Kierunek studiów: Informatyka

Specjalność: Systemy i technologie mobilne

Imię i nazwisko studenta: Florian Gaffke

Nr albumu: 175495

Poziom kształcenia: Studia drugiego stopnia

Forma studiów: niestacjonarne

Kierunek studiów: Informatyka

Specjalność: Systemy i technologie mobilne

PRACA DYPLOMOWA MAGISTERSKA

Tytuł pracy w języku polskim: Zastosowanie sztucznej inteligencji do tworzenia dialogów w grach wideo

Tytuł pracy w języku angielskim: The use of artificial intelligence to create dialogues in video games

Opiekun pracy: dr inż. Wioleta Szwoch

OŚWIADCZENIE dotyczące pracy dyplomowej zatytułowanej: **Zastosowanie sztucznej inteligencji do tworzenia dialogów w grach wideo**

Imię i nazwisko studenta: Karol Lange

Data i miejsce urodzenia: 07.06.1999, Świecie

Nr albumu: 175921

Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki

Kierunek: Informatyka

Poziom kształcenia: drugi

Forma studiów: niestacjonarne

Typ pracy: praca dyplomowa magisterska

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2019 r. poz. 1231, z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (t.j. Dz. U. z 2020 r. poz. 85, z późn. zm.),¹ a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

29.09.2024, Karol Lange

Data i podpis lub uwierzytelnienie w portalu uczelnianym Moja PG

**) Dokument został sporządzony w systemie teleinformatycznym, na podstawie §15 ust. 3b Rozporządzenia MNiSW z dnia 12 maja 2020 r. zmieniającego rozporządzenie w sprawie studiów (Dz.U. z 2020 r. poz. 853). Nie wymaga podpisu ani stempla.*

¹ Ustawa z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce:

Art. 312. ust. 3. W przypadku podejrzenia popełnienia przez studenta czynu, o którym mowa w art. 287 ust. 2 pkt 1–5, rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego.

Art. 312. ust. 4. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 5, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o podejrzeniu popełnienia przestępstwa.

STRESZCZENIE

Celem niniejszej pracy była analiza istniejących technologii z zakresu sztucznej inteligencji, w szczególności dużych modeli językowych, pod kątem ich wykorzystania w tworzeniu dialogów postaci niezależnych w grach wideo, a także zaprojektowanie i implementacja prototypowej gry, w której dialogi będą generowane przy pomocy narzędzi sztucznej inteligencji umożliwiających prowadzenie konwersacji w języku naturalnym. Gra wideo została wykonana za pomocą silnika do tworzenia gier Unity, a do obsługi dialogów wykorzystano duży model językowy GPT-4o stworzony i udostępniony przez firmę OpenAI. Ocena jakości dialogów generowanych przez model odbyła się na zasadzie badania użytkowników w formie ankiety wykonywanej po przejściu gry, a celem była ocena ich doświadczeń związanych z interakcjami z postaciami niezależnymi napędzanymi przez sztuczną inteligencję.

Karol Lange wykonał projekt wizualny gry, zaprojektował postacie, środowiska oraz ścieżkę dźwiękową i efekty dźwiękowe. Jest również autorem mechanik interakcji, poruszania się, systemu zadań, sztucznej inteligencji przeciwników i postaci niezależnych oraz systemu zarządzania audio. Zajmował się także zarządzaniem systemem kontroli wersji, serwerem Unity Cloud oraz testowaniem. Jest ponadto autorem rozdziałów 2.2, 2.3, 3.1, 3.2, 4.1, 4.2, 4.5, 5.1, 5.3, 5.5. Jest również współautorem rozdziałów 5.2 i 6.4.

Florian Gaffke był odpowiedzialny za zarządzanie projektem. Zaprojektował system dialogowy, system statystyk i zdrowia, system walki oraz system zarządzania interfejsem użytkownika. Był odpowiedzialny za koordynację działań, komunikację z interesariuszami, tworzenie interfejsu użytkownika oraz testowanie. Jest ponadto autorem rozdziałów 2.1, 2.4, 2.5, 3.3, 4.3, 4.4, 5.4, 6.1, 6.2, 6.3, 6.6. Jest również współautorem rozdziałów 5.2 i 6.4.

W badaniu uczestniczyło 41 osób, a wyniki wykazały, że generowanie dialogów przez model GPT pozytywnie wpłynęło na immersję i satysfakcję graczy. Dialogi były wysoko ocenione pod względem poprawności językowej i spójności z kontekstem gry. Niemniej jednak pojawiały się niespójności i błędy systemowe, które negatywnie wpływają na odbiór gry i ukończenie zadań, wskazując na potrzebę dalszych usprawnień, szczególnie w zakresie zarządzania mechaniką gry i filtrowania treści przez duży model językowy. Wykazano również, że gracze wciąż preferują tradycyjne formy interakcji, jak wybór gotowych opcji dialogowych, zamiast bardziej interaktywnych metod, takich jak użycie mikrofonu lub wpisywanie własnego tekstu.

Słowa kluczowe: sztuczna inteligencja, duże modele językowe, przetwarzanie języka naturalnego, systemy dialogowe, gry wideo

Dziedzina nauki i techniki, zgodnie z wymaganiami OECD: 1.2 Nauki o komputerach i informatyka

ABSTRACT

This thesis aims was to analyze existing artificial intelligence technologies, particularly large language models, for their use in creating non-player character dialogues in video games, and to design and implement a prototype game in which dialogues will be generated using artificial intelligence tools that enable natural language conversations. The video game was made using the Unity game development engine, and the GPT-4o large language model created and provided by OpenAI was used to manage the dialogues. The evaluation of the quality of the dialogues generated by the model was done by surveying users in the form of a questionnaire taken after playing the game, to assess their experience of interacting with artificial intelligence-powered non-player characters.

Karol Lange created the game's visual design and designed characters, environments, soundtrack, and sound effects. He is also the author of interaction mechanics, movement system, quest system, artificial intelligence for enemies and non-player characters, and the audio management system. He was also responsible for version control system management, Unity Cloud server, and testing. Moreover he is the author of chapters 2.2, 2.3, 3.1, 3.2, 4.1, 4.2, 4.5, 5.1, 5.3, 5.5. He is also co-author of chapters 5.2 and 6.4.

Florian Gaffke was responsible for project management. He designed the dialogue system, the health and statistics system, the combat system, and the user interface management system. He was responsible for coordinating activities, communicating with stakeholders, creating the user interface, and testing. Moreover, he is the author of chapters 2.1, 2.4, 2.5, 3.3, 4.3, 4.4, 5.4, 6.1, 6.2, 6.3, 6.6. He is also co-author of chapters 5.2 and 6.4.

The study involved 41 participants, and the results showed that generating dialogues with the GPT model positively impacted players' immersion and satisfaction. The dialogues were highly rated in terms of linguistic accuracy and consistency with the game context. However, there were inconsistencies and system errors that negatively affected the game experience and quest completion, indicating the need for further improvements, particularly in managing game mechanics and content filtering by the large language model. It was also shown that players still prefer traditional forms of interaction, such as selecting predefined dialogue options, rather than more interactive methods like using a microphone or typing their text.

Keywords: artificial intelligence, large language models, natural language processing, dialogue systems, video games

SPIS TREŚCI

Wykaz ważniejszych oznaczeń i skrótów	8
1. Wstęp i cel pracy	9
2. Problematyka niezależnych postaci w grach wideo	11
2.1. Niezależne postacie w grach	11
2.1.1. Wiarygodni NPC	11
2.1.2. Pojęcie charakteru i rozpoznawanie NPC	12
2.1.3. Procesy decyzyjne	12
2.1.4. Wyzwania związane z wiarygodnością NPC	12
2.1.5. Podsumowanie	13
2.2. Przetwarzanie języka naturalnego	13
2.2.1. Generowanie i rozumienie języka naturalnego	14
2.2.2. Systemy dialogowe	15
2.2.3. Systemy dialogowe w grach fabularnych	15
2.3. Modele generowania dialogów oparte na uczeniu głębokim	15
2.3.1. Sieci neuronowe	16
2.3.2. Transformer i mechanizm uwagi	17
2.3.3. Duże modele językowe	19
2.4. Zaawansowane techniki generowania dialogu dla rozwoju postaci NPC	19
2.4.1. Drzewa dialogowe i interaktywna narracja	20
2.4.2. Systemy punktacji i atrybutów	20
2.4.3. Generowanie dialogów na podstawie AI	20
2.4.4. Lokalizacja i adaptacja kulturowa	21
2.4.5. Wyzwania i ograniczenia	21
2.4.6. Przyszłe kierunki rozwoju	21
2.4.7. Podsumowanie	22
2.5. Metody oceny jakości generowanych dialogów	22
2.5.1. Ewaluacja automatyczna	22
2.5.2. Ewaluacja heurystyczna	23
2.5.3. Ocena użytkowników	24
2.5.4. Samoocena	24
2.5.5. Podsumowanie	24
3. Technologie, algorytmy i narzędzia	26
3.1. GPT	26
3.1.1. Proces uczenia GPT	26
3.1.2. GPT-4	26
3.1.3. Interfejs programistyczny OpenAI	27
3.2. Unity	29
3.2.1. Architektura Unity	30
3.2.2. Edytor i narzędzia	30
3.2.3. Unity Cloud	31

3.3. Język programowania i środowisko programistyczne	32
3.3.1. Język C#	32
3.3.2. Środowisko Visual Studio	32
4. Specyfikacja i analiza wymagań	34
4.1. Specyfikacja wymagań	34
4.1.1. Cel i przeznaczenie gry	34
4.1.2. Kontekst realizacji	34
4.1.3. Charakterystyka gry	34
4.1.4. Środowisko pracy i sposób rozpowszechniania	34
4.1.5. Ograniczenia	35
4.2. Wymagania funkcjonalne	35
4.2.1. Dialogi	35
4.2.2. Postacie niezależne	35
4.2.3. Treść i świat gry	36
4.2.4. Dźwięk	36
4.2.5. Zadania	36
4.2.6. Statystyki i system walki	36
4.2.7. Inne funkcje	36
4.3. Wymagania niefunkcjonalne	37
4.3.1. Wydajność	37
4.3.2. Kompatybilność	37
4.3.3. Użyteczność	37
4.3.4. Niezawodność	37
4.3.5. Przechowywanie danych	37
4.3.6. Integracja z zewnętrznymi systemami	38
4.3.7. Bezpieczeństwo i prywatność	38
4.4. Kryteria akceptacji	38
4.4.1. Testy alfa	38
4.4.2. Walidacja promotora oraz graczy	39
4.4.3. Testy beta	39
4.5. Zespół projektowy	40
5. Projekt gry GPT Chronicles	41
5.1. Architektura gry	41
5.1.1. Diagram architektury	41
5.1.2. Diagram pakietów	42
5.1.3. Diagram komponentów składowych	42
5.1.4. Struktura projektu	46
5.2. Mechaniki gry	48
5.3. Projekt Unity	71
5.4. Projekt interfejsu użytkownika	75
5.5. Projekt danych	81
5.5.1. Dialogi	81
5.5.2. Zadania	86
5.5.3. Ankiety	87

6. Badania.....	91
6.1. Cel badań i jego uzasadnienie	91
6.2. Problemy badawcze	91
6.3. Organizacja i przebieg badań	91
6.3.1. Wstępna ocena poprawności i jakości dialogów oraz walidacja wyników	91
6.3.2. Projekt ankiety	92
6.3.3. Przebieg ankietyzacji	96
6.4. Wyniki	96
6.4.1. Charakterystyka demograficzna respondentów	97
6.4.2. Ocena interakcji z grą	99
6.5. Oceny ilościowe	99
6.5.1. Ocena interakcji z NPC	99
6.5.2. Ocena jakości systemu	100
6.5.3. Ocena dialogów	101
6.5.4. Ocena klarowności gry	102
6.5.5. Satysfakcja i zaangażowanie	103
6.5.6. Błędy i niespójności w dialogach	103
6.5.7. Analiza korelacji	104
6.5.8. Analiza jakościowa	105
6.5.9. Trudności napotykane przez graczy	105
6.5.10. Satysfakcjonujące aspekty interakcji z AI	106
6.6. Samoocena	106
6.6.1. Etap 1: Analiza dialogów i wyników zadań	106
6.6.2. Etap 2: Ocena jakościowa dialogów przez LLM	108
7. Wnioski	112
Wykaz literatury	116
Spis rysunków	118
Spis tabel	119

WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

AI (*ang. artificial intelligence*) – sztuczna inteligencja

API (*ang. application programming interface*) – interfejs programistyczny

AR (*ang. augmented reality*) – rzeczywistość rozszerzona

D&D (*ang. Dungeons & Dragons*) – Lochy i Smoki: popularna gra fabularna, w której gracze wcielają się w postacie w fikcyjnym świecie, odgrywając ich rolę i tworząc opowieść pod przewodnictwem mistrza gry (*ang. Dungeon Master*)

DL (*ang. deep learning*) – uczenie głębokie

GPT (*ang. generative pre-trained transformer*) – wstępnie przeszkolony transformator generatywny

IDE (*ang. integrated development environment*) – zintegrowane środowisko programistyczne

LLM (*ang. large language model*) – duży model językowy

LSTM (*ang. long short-term memory*) – długoterminowa pamięć krótkoterminowa

MMORPG (*ang. massively multiplayer online role-playing game*) – gra fabularna online dla wielu graczy

NLG (*ang. natural language generation*) – generowanie języka naturalnego

NLP (*ang. natural language processing*) – przetwarzanie języka naturalnego

NLU (*ang. natural language understanding*) – rozumienie języka naturalnego

NPC (*ang. non-player character*) – bohater niezależny

RNN (*ang. recurrent neural network*) – rekurencyjna sieć neuronowa

RPG (*ang. role-playing game*) – gra fabularna

VR (*ang. virtual reality*) – rzeczywistość wirtualna

1. WSTĘP I CEL PRACY

W ostatnich latach obserwujemy dynamiczny rozwój sztucznej inteligencji AI (*ang. artificial intelligence*) związanej z generacją języka naturalnego NLG (*ang. natural language generation*) i dużych modeli językowych LLM (*ang. large language model*). Technologie te stają się coraz bardziej zaawansowane, co prowadzi do rozwoju nowych zastosowań w różnych dziedzinach, takich jak edukacja, rozrywka, biznes i administracja. Szczególnym przykładem jest ChatGPT - narzędzie opracowane przez firmę Open AI, które niedawno zyskało rozgłos w mediach. Ono jako pierwsze wzbudziło tak szerokie zainteresowanie opinii publicznej, zwiększaając świadomość potencjału, możliwości, ale także zagrożeń związanych ze sztuczną inteligencją. Jednak mimo rewolucji jaką wydaje się wprowadzać przetwarzanie języka naturalnego NLP (*ang. natural language processing*) we wszystkich dziedzinach nauki i technologii, przemysł tworzenia gier video wciąż pozostaje poza zasięgiem tych innowacji. Mogłoby się wydawać, że potencjał jaki posiada NLG mógłby zrewolucjonizować sposób tworzenia dialogów dla bohaterów niezależnych NPC (*ang. non-player character*), jednak efektywność takich systemów w budowaniu kreatywnych i spójnych linii dialogowych w grach video wciąż wymaga analizy i dogłębnych badań.

Tradycyjni NPC posiadający z góry ustalone linie dialogowe oferują bardzo ograniczone możliwości zaangażowania graczy, co jest szczególnie widoczne w grach z dużym otwartym światem z takich gatunków jak gry fabularne RPG (*ang. role-playing game*) czy również dla dużej liczby graczy MMORPG (*ang. massively multiplayer online role-playing game*), gdzie czas i środki, jakie należały poświęcić na produkcję ciekawych i angażujących dialogów dla tak dużej ilości NPC, znacząco przekracza możliwości nawet największych firm w branży. Dlatego też obecnie widać duże zainteresowanie generatywnym AI wśród badaczy i wiodących firm oraz możliwościami jego zastosowania w tworzeniu gier. Należy jednak pamiętać, że opracowanie takiego systemu wiąże się z kilkoma kwestiami, na które należy zwrócić uwagę. Musimy, więc zadać sobie następujące pytania:

- Jak zapewnić, że linie dialogowe są odpowiednie dla kontekstu gry?
- Jak zapewnić, że linie dialogowe są spójne z osobowością i charakterem NPC?
- Jak uniknąć generowania dialogów, które są obraźliwe lub szkodliwe?
- Jak zapewnić, że wygenerowane dialogi umożliwiają graczom odpowiednie zanurzenie się w świecie gry i wzmacniają ich więź z historią i napotkanymi postaciami?

W związku z tym konieczne jest również stworzenie odpowiednich metod oceny czy systemy NLG mogą zapewnić na tyle dużą jakość, aby osiągnąć poczucie zaangażowania u gracza poprzez wygenerowany przez nie dialog.

Celem niniejszej pracy jest udzielenie odpowiedzi na wcześniej zadane pytania, dokonanie kompleksowej analizy istniejących technologii z zakresu sztucznej inteligencji, ze szczególnym uwzględnieniem dużych modeli językowych oraz ocena ich potencjalnego zastosowania w tworzeniu dialogów dla NPC w grach video. Następnie stworzenie prototypowej gry fabularnej, w której dialogi będą generowane przy pomocy narzędzi sztucznej inteligencji oraz ocena jakości tych dialogów i doświadczeń związanych z interakcją z takimi NPC. Weryfikacja generowanych dialogów oraz ich ocena przez graczy pozwoli określić, w jakim stopniu sztuczna inteligencja może

zastąpić tradycyjne metody tworzenia dialogów w tworzeniu interaktywnych doświadczeń w grach wideo.

Praca została podzielona na kilka rozdziałów. W rozdziale 2. przedstawiono wstęp teoretyczny. Rozpoczyna się on od omówienia problematyki postaci niezależnych w grach wideo (2.1). Szczególny nacisk położono na wyzwania związane z przedstawieniem postaci w sposób wiarygodny i realistycznie odwzorowujący charakter, zachowanie i sposób podejmowania decyzji podobny do ludzkiego. Następnie przedstawiono zagadnienia związane z przetwarzaniem języka naturalnego przez systemy komputerowe (2.2), w tym generowanie i rozumienie mowy oraz systemy dialogowe. Praca opisuje również modele generowania dialogów oparte na uczeniu głębokim (2.3), ze szczególnym uwzględnieniem architektury transformer oraz dużych modeli językowych. Omówiono także współczesne zaawansowane techniki generowania dialogu dla postaci niezależnych (2.4), takie jak drzewa dialogowe, systemy punktacji i atrybutów oraz przyszłe kierunki rozwoju. Na koniec przedstawiono możliwe sposoby oceny jakości wygenerowanego dialogu (2.5).

W części projektowo-badawczej pracy zaprezentowano projekt i implementacje gry GPT Chronicles. Rozdział 3. rozpoczęto od szczegółowego opisu wykorzystanych narzędzi, technologii i algorytmów. W rozdziale 4. przedstawiono specyfikację techniczną wraz z analizą wymagań, w tym wymagania funkcjonalne (4.2), niefunkcjonalne (4.3) oraz kryteria akceptacji (4.4). Rozdział 5. zawiera dokładny opis architektury gry, jej struktury (5.1), mechanik wraz z przykładami implementacji (5.2), projekt interfejsu użytkownika (5.4) oraz struktury danych dla dialogów i zadań (5.5). Podział prac nad częścią praktyczną pracy został opisany w podrozdziale 4.5. Zespół projektowy.

Przeprowadzono także badania ankietowe wśród graczy, których celem była ocena skuteczności zastosowanych technik sztucznej inteligencji w tworzeniu wiarygodnych postaci niezależnych. W rozdziale 6. przedstawiono szczegółowy projekt ankiety, z podziałem na odpowiednie kategorie, oraz opisano przebieg samej ankietyzacji i procesu zbierania danych. Zebrane dane posłużyły do szczegółowej analizy zarówno problemów napotkanych przez graczy, jakości generowanych dialogów jak i samej satysfakcji i zaangażowania graczy. Następnie zaprezentowano wyniki tych badań oraz wnioski płynące z przeprowadzonej analizy.

2. PROBLEMATYKA NIEZALEŻNYCH POSTACI W GRACH WIDEO

Od zarania dziejów, dialog i komunikacja były kluczowymi elementami, które odróżniały człowieka od innych gatunków. To właśnie nasza zdolność do prowadzenia rozmów, wymiany myśli i idei, pozwoliła nam na rozwój i postęp. Język, w swojej najczystszej formie, jest nie tylko narzędziem komunikacji, ale także wyrazem naszej tożsamości, kultury i historii.

Gry komputerowe to jedynie kolejny obszar, w którym dialogi odgrywają kluczową rolę. W grach, dialogi są nie tylko narzędziem do przekazywania informacji o fabule czy zadaniach, ale również sposobem na budowanie relacji z postaciami niezależnymi, na wprowadzanie gracza w świat gry i na tworzenie emocjonalnej więzi. Dialog w grach komputerowych pozwala graczom na wczucie się w swoje postacie, na zrozumienie ich motywacji i celów. Dzięki temu, gracz może doświadczyć historii z perspektywy swojej postaci, co zdecydowanie wpływa na jakość doświadczenia z gry.

Systemy dialogowe w grach komputerowych stają się coraz bardziej zaawansowane, umożliwiając twórcom gier na tworzenie coraz bardziej realistycznych i angażujących doświadczeń. Dzięki temu, gracze mogą doświadczyć pełnej gamy emocji. Sztuczna inteligencja otwiera nowe możliwości dla systemów dialogowych. Dzięki AI, dialogi mogą stać się dynamiczne i interaktywne na niespotykaną dotąd skalę. Każda postać niezależna w grze, niezależnie od jej roli, mogłaby prowadzić angażujące i unikalne rozmowy z graczem, każda postać mogłaby mieć swoją unikalną osobowość, styl mówienia i historię do opowiedzenia. To z kolei pozwoliłoby graczom na nawiązanie silniejszych emocjonalnych więzi z postaciami i lepsze zrozumienie świata gry.

2.1. Niezależne postacie w grach (Gaffke)

W kontekście rozwoju gier wideo, postacie sterowane przez komputer, znane jako postacie niezależne od gracza NPC (*ang. non-player character*), są niezbędnym elementem w tworzeniu zanurzenia (*ang. immersion*) i dynamiki świata gry. Postacie te, zarządzane przez skomplikowane algorytmy, a coraz częściej przez sztuczną inteligencję, nie tylko przyczyniają się do stworzenia odpowiedniego klimatu gry, ale mogą również aktywnie wchodzić w interakcje z graczem, pełniąc różne role - od wrogów przez dostawców zasobów, po kompanów w pokonywaniu fabuły. Wykorzystanie zaawansowanej sztucznej inteligencji (AI) pozwala na zarządzanie NPC w sposób, który przyczynia się do budowania atmosfery i wspiera narrację gry, niezależnie od ich bezpośredniego udziału w głównym wątku fabularnym. Oczekuje się od nich, aby działały płynnie i przemyślanie, bez powodowania zakłóceń w rozgrywce, a ich nieodpowiednie działanie może prowadzić do frustracji wśród graczy [1].

2.1.1. Wiarygodni NPC

Projektowanie przekonujących NPC wymaga skupienia się na kilku fundamentalnych aspektach, wśród których kluczową rolę odgrywa spójność ich zachowań z oczekiwaniami graczy oraz z założeniami świata przedstawionego w grze. Wyzwaniem jest zapewnienie, aby NPC nie byli zbyt przewidywalni lub nie do pokonania, co mogłoby zniechęcać graczy [2] [3]. Dlatego kluczowe jest zaprogramowanie NPC z zachowaniami, które są zróżnicowane i dostosowane do kontekstu

gry, wzbogacone o symulację emocji, co według badań [3] [4] zwiększa zaangażowanie graczy i wpływa na ich decyzje.

2.1.2. Pojęcie charakteru i rozpoznawanie NPC

Rozpoznawanie NPC oraz zrozumienie ich funkcji ogrywają istotną rolę w kształtowaniu doświadczeń graczy, wpływając na interakcję i immersję w grze. Wygląd i zachowanie NPC dostarczają kluczowych informacji o możliwościach interakcji. Warpefelt [2] wprowadził pojęcie "charakteru" dla NPC, które aktywnie uczestniczą w grze i komunikują swoje cele. Charakter opiera się na sześciu aspektach osobowości zidentyfikowanych przez Dennetta (1981): racjonalności, intencjonalności, mowie, postawie, interakcji, komunikacji i świadomości. Te elementy, choć nie zawsze wymagane dla każdego NPC, muszą tworzyć spójny i przekonujący obraz. NPC osiągają "charakter", gdy mają swoje cele i aktywnie wpływają na grę. Ich rola i funkcje w grze określają głębię charakteru. Warpefelt [2] zidentyfikował trzy kryteria, dzięki którym gracze mogą rozpoznać role NPC: środowisko i lokalizacja, interaktywność oraz atrybuty i komunikacja wizualna. Te wskaźniki umożliwiają dedukcję tożsamości, charakteru i zachowań NPC.

Innym istotnym aspektem mającym wpływ na doświadczenia z postaciami niezależnymi jest ich zdolność do rozwoju i adaptacji wraz z postępem gry. Oznacza to, że NPC powinni być świadomi i reagować na zmiany następujące w świecie gry, posiadać wiedzę o wcześniejszych interakcjach z graczem lub światem, żeby móc się do nich odnosić w przyszłych rozmowach, a także posiadać własne rutyny i cele, które zmieniają się w odpowiedzi na decyzje i działania gracza. Dynamiczna natura NPC powoduje wzmacnianie charakteru, a tym samym jest przyczynkiem do pogłębienia immersji gracza, ponieważ takie zachowania powodują wzrost realizmu świata przedstawionego, w którym decyzje i działania gracza mają zauważalny wpływ na otoczenie.

Podkreślając znaczenie środowiska, wizualnych oraz behawioralnych wskaźników w zrozumieniu ról NPC, nie można pominać zdolności do adaptacji i ewolucji. Wiarygodność postaci NPC, wspierana przez te kryteria, jest kluczowa dla utrzymania immersji gracza w świecie gry.

2.1.3. Procesy decyzyjne

Proces podejmowania decyzji przez NPC opiera się na hierarchicznych strukturach, gdzie akcje są podzielone na węzły w grafie decyzyjnym, kierujące zachowaniami postaci w scenariuszach. Zmiany wywołane przez czynniki zewnętrzne, takie jak ocena środowiska, nagrody, kary i wpływ emocjonalny determinują przejście z jednego węzła do drugiego [3]. Efektywność tego procesu jest kluczowa dla realistyczności postaci, a jego złożoność może wpływać na wiarygodność NPC. Prostsze procesy podejmowania decyzji skutkują bardziej sztucznyimi decyzjami, podczas gdy bardziej złożone procesy wymagają cech, takich jak skalowalność i indeterminizm [3].

2.1.4. Wyzwania związane z wiarygodnością NPC

Zaprojektowanie wiarygodnych postaci NPC jest dużym wyzwaniem w branży gier. Postacie te są uznawane za wiarygodne, kiedy ich działania odpowiadają ustalonym schematom, jednak ich interakcje z graczami są ograniczone przez możliwości zaprogramowane przez twórców. Niespodziewane ruchy graczy mogą zakłócić tę wiarygodność, co skłania twórców do zawężania dostępnych opcji w celu uniknięcia odbioru gry jako sztucznej. Konwencjonalne metody tworzenia sztucznej inteligencji dla NPC często komplikują zadanie osiągnięcia realistycznych zachowań. Wzrastające oczekiwania wobec zarządzania zachowaniami AI NPC, ze względu na standardy

jakości, wskazując na konieczność zmian w procesach produkcyjnych, by zapewnić zaawansowane zachowania poprawiające wrażenia z gry [2] [3].

2.1.5. Podsumowanie

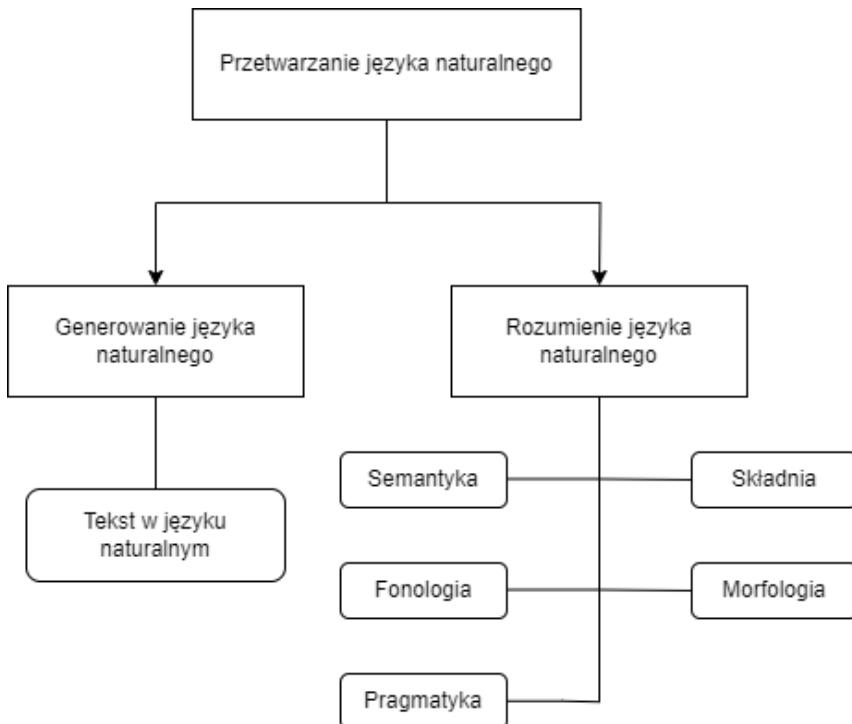
Wyzwania związane z projektowaniem wiarygodnych NPC wynikają głównie z ograniczeń interakcji z graczami i konieczności szczegółowego programowania ich zachowań. Osiągnięcie wysokiego poziomu wiarygodności postaci wymaga zastosowania nowoczesnych metod i technologii w projektowaniu AI, co jest przedmiotem badań i rozwoju w branży gier wideo [1]. Tradycyjne metody mogą okazać się niewystarczające, co skłania do poszukiwania innowacyjnych rozwiązań w celu poprawy realizmu i wpływu NPC na doświadczenie graczy.

2.2. Przetwarzanie języka naturalnego (Lange)

Przetwarzanie języka naturalnego NLP (*ang. natural language processing*) to dziedzina informatyki i sztucznej inteligencji, a także językoznawstwa, która rozwija systemy implementujące rozumienie języka naturalnego [5]. NLP skupia się na analizie, zrozumieniu, generowaniu i przetwarzaniu tekstu lub mowy w języku naturalnym. Obejmuje wiele aspektów, w tym analizę leksykalną, składniową i semantyczną, rozpoznawanie mowy, przekład maszynowy, analizę sentymetu czy generowanie tekstu. Nie jest to trywialne zadanie, ponieważ języki to niezwykle złożone systemy reguł gramatycznych, semantycznych i pragmatycznych, a te same słowa czy zdania mogą mieć wiele znaczeń, co znacząco utrudnia jednoznaczne zrozumienie kontekstu i sensu komunikatu. Sam kontekst również jest związany z kulturą i społeczeństwem, co sprawia, że jego zrozumienie może wymagać wiedzy na tematy kultury i historii danego społeczeństwa. Kolejnym istotnym aspektem generowania i interpretacji jest struktura morfologiczna języków, czyli sposobu, w jaki słowa są tworzone i zmieniane, aby wyrażać różne znaczenia gramatyczne. W ten sposób możemy wydzielić dwa główne typy: języki analityczne i języki syntetyczne. Przykładem języka należącego do pierwszej grupy jest współczesny język angielski, który wyraża funkcje gramatyczne za pomocą luźnych morfemów (najmniejsza cząstka wyrazu) w postaci np. przyimków, elementów przysłówkowych, zaimkowych oraz słów posiłkowych [6]. Druga grupa jest przeciwieństwem i reprezentuje ją między innymi język polski. Są to języki, w których stosunki gramatyczne w zdaniu oznaczane są w formach wyrazów za pomocą afiksów, łączących kilka funkcji, które występują w charakterze pojedynczego słowa [7]. Dla przykładu, afiksy w języku polskim są szeroko stosowane m.in. do tworzenia słów i form gramatycznych:

- Odmiana przez przypadki: dom, domu, domowi, domem
- Tworzenie liczby mnogiej: kot, koty
- Stopniowanie przymiotników: mądry, mądrzejszy, najmądrzejszy
- Słowotwórstwo: nosić listy, listonosz

NLP może zostać podzielone na dwie zasadnicze podkategorie tzn. generowania języka naturalnego i rozumienie języka naturalnego, czyli zadania opierające się kolejno na próbie zrozumienia podanego tekstu i jego generowaniu, zostały one przedstawione na rys. 2.1.



Rysunek 2.1: Podział NLP

2.2.1. Generowanie i rozumienie języka naturalnego

Generowanie języka naturalnego NLG (*ang. natural language generation*) odnosi się do obszaru sztucznej inteligencji zajmującego się automatycznym tworzeniem tekstu lub mówionego języka naturalnego. NLG jest, więc procesem, w którym komputer lub inny system informatyczny przekształca dane wejściowe, często w formie danych strukturalnych lub informacji, na zrozumiałą i czytelny tekst w języku naturalnym, takim jak angielski, polski czy inny język. Wykorzystuje więc wiedzę o języku, aby w sposób automatyczny produkować dokumenty, raporty, wyjaśnienia, dialogi i wszystkie inne rodzaje form pisma [8]. Natomiast jak zauważają autorzy pracy [9] podana definicja, mimo iż jest jedną z najczęściej cytowanych w tej dziedzinie i wszyscy wydają się być zgodni co do tego, jaki powinien być wynik systemu NLG (tekstowy), to zdaniem wielu autorów jest nie do końca precyzyjna w kwestii danych wejściowych, a te mogą się znacznie różnić. Obejmują one między innymi dane numeryczne, ustrukturyzowane bazy wiedzy, a obecnie ważnym wyzwaniem NLG stało się generowanie tekstu na podstawie danych wizualnych, takich jak obrazy lub wideo.

Kolejnym podzbiorem NLP jest rozumienie języka naturalnego NLU (*ang. natural language understanding*). NLU odnosi się do analizy składniowej (*ang. syntax*), semantycznej, a także pragmatycznej tekstu w celu określenia znaczenia danego zdania. Przed przystąpieniem do tych analiz niezbędne jest przeprowadzenie wcześniejszych etapów, takich jak analiza leksykalna, rozpoznanie fonemów, przetwarzanie wstępne oraz segmentacja. Składnia odnosi się do struktury gramatycznej zdania, semantyka do jego zamierzonego znaczenia, a pragmatyka bierze pod uwagę kontekst i zamierzenia mówcy, uwzględniając także indywidualny styl wypowiedzi. NLU tworzy również odpowiednią strukturę danych, która umożliwia określenie relacji między słowami i frazami, uwzględniając kontekst i inne elementy analizy. Aby komputer zrozumiał zamierzone znaczenie różnych tekstów, które przychodzi ludziom naturalnie podczas rozmowy, wymagana jest kombinacja tych analiz. Odpowiednie zrozumienie kontekstu zdania jest niezbędne do wygenerowania

odpowiedzi mogącej w sposób realistyczny imitować odpowiedź jakiej spodziewa się człowiek.

2.2.2. Systemy dialogowe

System dialogowy to program komputerowy mający na celu interakcje z użytkownikami w formie dialogu w języku naturalnym. Zgodnie z zastosowaniem systemy dialogowe można podzielić w zasadzie na dwie grupy tj. systemy zorientowane zadaniowo (*ang. task-oriented*) oraz systemy niezorientowane na zadania (znane również jako chatboty) [10]. Systemy zorientowane zadaniowo mają na celu wesprzeć użytkownika w wykonaniu pewnych zadań takich jak na przykład rekomendowanie produktów, rezerwacja biletów. Oznacza to, że są one zoptymalizowane pod kątem określonego zakresu działania w przeciwieństwie do systemów niezorientowanych zadaniowo, które są otwarte na różne rodzaje konwersacji i niekoniecznie muszą rozwiązywać potrzeby użytkownika. Ich celem jest zapewnienie rozrywki i towarzystwa użytkownikowi. Zarówno w jednym jak i drugim typie wykorzystuje się techniki NLP do rozumienia i generowania tekstu. W ostatnich latach podstawową techniką w systemach dialogowych stało się uczenie głębokie DL (*ang. deep learning*), które wypiera standardowe modele oparte o reguły. Sieci neuronowe są stosowane w różnych komponentach tradycyjnego systemu dialogowego, w tym NLU, NLG, a także śledzenia stanu dialogu. Wykorzystanie olbrzymiej ilości danych umożliwia bowiem zbudowania ujednoliconego inteligentnego systemu dialogowego, w którym zacierają się granice między systemami zorientowanymi zadaniowo i systemami otwartymi [10]. Co więcej neuronowe systemy dialogowe mają duży potencjał w imitacji ludzkich emocji podczas prowadzenia przez nie konwersacji, a także w utrzymywaniu kontekstu rozmowy, w którym komputer zachęca użytkownika do rozmowy mającej długoterminowy cel. Są to cechy niezwykle pożądane w dialogach w grach wideo, gdzie chatbot mógłby poprowadzić gracza przez konkretną narrację z uwzględnieniem odpowiednich emocji takich jak gniew, radość czy smutek.

2.2.3. Systemy dialogowe w grach fabularnych

Systemem dialogowym w kontekście gry wideo nazywamy mechanikę gry, która tekstowo reprezentuje rozmowę pomiędzy graczem a co najmniej jedną postacią niezależną NPC (*ang. non-playable character*) [11]. Gracz podczas interakcji z NPC zyskuje możliwość przeprowadzenia z nim rozmowy poprzez decydowanie, co zostanie im powiedziane przez jego postać. W grach fabularnych dialogi są predefiniowane, gracz w takim przypadku wybiera jedną z dostępnych opcji dialogowych w danej turze. Niektóre gry wprowadzają systemy dynamiczne poprzez tzw. rozgałęzienia, gdzie kierunek rozmowy zależny jest od wcześniejszych wyborów jakie podjął gracz. Oznacza to, że dialog może zakończyć się na różne sposoby, aby dodatkowo zwiększyć imersję i poczucie zaangażowania gracza, a jego wybory mają bezpośredni wpływ na świat gry i fabułę. Tworzenie dialogów w grach jest, więc procesem bardzo złożonym, w szczególności, jeśli twórcy gry chcą zapewnić duży poziom interaktywności i zawartości opcjonalnej w postaci misji pomocniczych. Najlepszym przykładem jest niedawno wydana gra Baldur's Gate III [12], w której skrypt zawiera oszałamiające dwa miliony słów [13]. Nie trzeba, więc dodawać, że dialogi są niezwykle istotnym aspektem tworzenia gier, który może znacząco wydłużyć proces produkcji.

2.3. Modele generowania dialogów oparte na uczeniu głębokim (Lange)

Techniki uczenia maszynowego umożliwiają budowanie dynamicznych systemów NLG, które są w stanie generować znacznie bardziej skomplikowane teksty bez potrzeby wcześniejszego de-

finiowania tekstu w postaci reguł, z których maszyna musiałaby wybierać. Najprostszym takim systemem są łańcuchy Markova (*ang. Markov chains*). łańcuch Markova to model probabilistyczny, który opisuje prawdopodobieństwo wystąpienia danego zdarzenia na podstawie wydarzeń, które wystąpiły wcześniej. W przypadku generowania tekstu łańcuch Markova jest używany do przewidywania kolejnego słowa w zdaniu na podstawie słów, które już wystąpiły. Na przykład, jeśli mamy zdanie "Ktoś wszedł do pokoju", to łańcuch Markova na podstawie wcześniejszego uczenia na pewnym zbiorze danych może przewidzieć, że następnym słowem będzie "i" z prawdopodobieństwem 0,25. Jest to prawdopodobieństwo, że słowo "i" wystąpi po słowie "pokoju" w zbiorze danych, na którym został przeszkolony model. łańcuch Markova jest prosty jednak przez to, że nie zapamiętuje stanu poprzedniego słowa, a przewiduje jedynie prawdopodobieństwo, co skutkuje powtarzalnością jego odpowiedzi. Dodatkowo nie może używać słów, których nie było w zbiorze danych treningowych, ponieważ jest w stanie budować odpowiedzi tylko na słowach znajdujących się w tym zbiorze, bo tylko dla nich jest w stanie przewidzieć prawdopodobieństwo [14].

2.3.1. Sieci neuronowe

Rekurencyjna sieć neuronowa RNN (*ang. recurrent neural network*) jest próbą imitacji procesów zachodzących w ludzkim mózgu. Jest to sieć, która może przetwarzać dane w sposób sekwencyjny [15]. Oznacza to, że RNN mogą zapamiętywać stan poprzedniego słowa lub fragmentu tekstu i używać tej informacji do generowania następnego słowa lub fragmentu tekstu. Nazywamy to pamięcią krótkoterminową (*ang. short-term memory*). RNN są często używane do generowania tekstu, w tym dialogów. Aby wygenerować dialog, należy podać początkowy fragment tekstu, a następnie RNN będzie generował kolejne fragmenty na podstawie poprzednich. RNN może generować złożone i naturalne dialogi, jeśli zostanie przeszkolony na dużych zbiorach danych. Jednak przez to, że posiada jedynie pamięć krótkoterminową, dłuższe wypowiedzi zaczynają pod koniec tracić spójność i kontekst ze względu na zanik pamięci.

Długa pamięć krótkoterminowa LSTM (*ang. long short-term memory*) została zaproponowana w 1997 roku przez Hochreiter'a i Schmidhuber'a[16] jako rozwiązywanie problemu zanikającej pamięci w RNN. LSTM przypominają standardowe RNN, ale tutaj każdy węzeł sieci zostaje zastąpiony komórką pamięci, a każda z nich zawiera stan wewnętrzny. LSTM może odczytywać, zapisywać i usuwać informacje z tej pamięci poprzez wprowadzenie bramki wejściowej (*ang. input gate*), bramki zapomnienia (*ang. forget gate*) i bramki wyjściowej (*ang. output gate*), które decydują ile informacji z nowych danych wejściowych i wspomnień z przeszłości należy zapisać. Oznacza to, że ten model może zastosować pamięć długoterminową jak i krótkoterminową do kodowania danych sekwencyjnych i wykorzystuje mechanizmy bramek, aby kontrolować przepływ informacji.

Kiedy chcemy zbudować realistyczny model mogący reprezentować postacie niezależne pamięć jest kluczowym elementem, możliwość odwoływanego się do wiedzy zewnętrznej i doświadczeń, a nie tylko do ostatnich wypowiedzi jest kluczowe w budowaniu wiarygodnych bohaterów. LSTM mimo, że odnosi ogromne sukcesy w wielu dziedzinach to wciąż posiada za mały moduł pamięci przez co ciężkie jest skompresowanie konkretnych faktów i wiedzy do ich ponownego wykorzystania w przyszłości. W 2015 roku Jason Weston zaproponował, więc sieci pamięci (*ang. memory networks*) [17] czyli model wyposażony w całkowicie oddzielny moduł pamięci długoterminowej. Sieć pamięci składa się zasadniczo z pięciu modułów:

- "I"(*ang. input feature map*) - konwertuje dane wejściowe na reprezentacje cech
- "G"(*ang. generalization*) - aktualizuje stare wspomnienia, biorąc pod uwagę nowe dane wejściowe

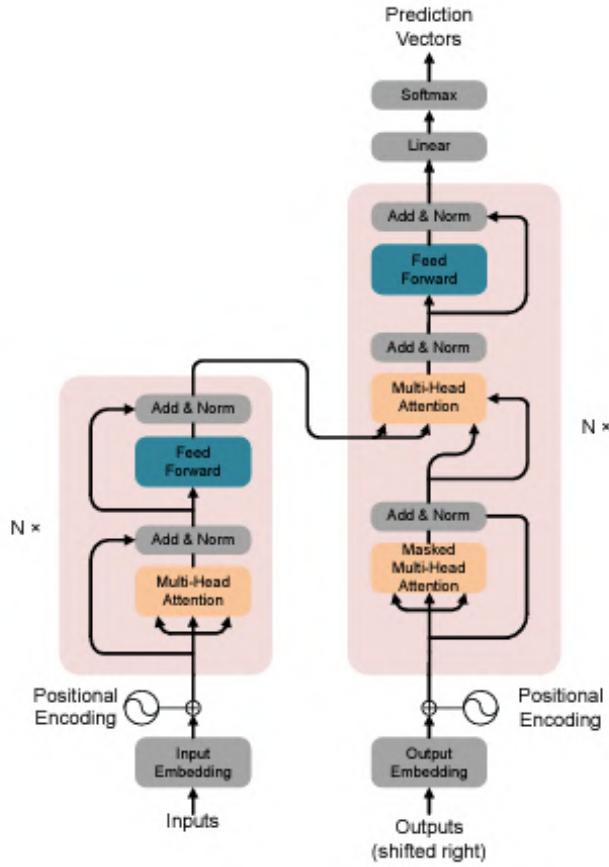
- "O"(ang. *output feature map*) - produkuje nowe dane wyjściowe (w przestrzeni reprezentacji cech), biorąc nowe dane wejściowe i obecny stan pamięci
- "R"(ang. *response*) - przekształca dane wyjściowe na wynik tekstowy
- "M"(ang. *memory*) - pamięć, która przechowuje reprezentacje faktów

Model ten wymaga uczenia nadzorowanego dla każdego modułu z osobna, dlatego w [18] zaproponowano usprawnienie tego modelu poprzez trenowanie end-to-end, które znaczająco ułatwiło uczenie i ich praca jest uznawana za bazową sieć pamięci.

Wiele prac związanych z generowaniem dialogów wprowadza sieć pamięci (*ang. memory networks*) lub jej wariacje jako jedno z rozwiązań, w szczególności w systemach dialogowych zorientowanych zadaniowo wymagających wiedzy zewnętrznej. W [19] wykorzystano sieci pamięci dla systemów wieloturowych systemów dialogowych, które wymagają utrzymania kontekstu w pamięci przez długi czas. Zaproponowano model DDMN (*ang. dual dynamic memory network*), w którym sieć składa się z dwóch komponentów pamięci: menedżer pamięci dialogu i menedżer pamięci bazy wiedzy. Menedżer pamięci dialogowej dynamicznie rozszerza pamięć dialogową w każdej turze dialogu i śledzi historię dialogową za pomocą mechanizmu aktualizacji, który zachęca model do filtrowania nieistotnej historii dialogowej i zapamiętywania ważnych, nowo pojawiających się informacji. Menedżer pamięci bazy wiedzy natomiast udostępnia informacje w całej konwersacji i dynamicznie wyodrębnia dane za pomocą wskaźnika pamięci na każdym kroku. Badacze uzyskali znaczący wzrost jakości dialogu względem innych modeli porównawczych zarówno w ewaluacji automatycznej jak i wykonanej przez ludzi [19]. Kolejny model (WMM2Seq) wykorzystujący sieci pamięci zaproponowali autorzy [20]. Stworzyli oni system dialogowy zorientowany zadaniowo składający się aż z trzech modułów pamięci: dwóch modułów pamięci długoterminowej przechowujących odpowiednio historię dialogu i bazę wiedzy oraz moduł pamięci roboczej przeprowadzający z nimi interakcje i zawierający pamięć epizodyczną. Ten model również uzyskał wyniki przewyższające klasyczne modele [20].

2.3.2. Transformer i mechanizm uwagi

Mechanizm uwagi (*ang. attention*) zaproponowano w 2014 roku w pracy [21] jako poprawę dla neuralnego tłumaczenia maszynowego. Autorzy zaproponowali sposób na lepsze dopasowanie słów, które są ze sobą powiązane w różnych językach. Istnieje kilka różnych sposobów jego implementacji. Jednym z najpopularniejszych jest użycie sieci neuronowej do obliczania współczynników uwagi dla każdej części danych. Te współczynniki wskazują, jak ważna jest część danych dla generowanej odpowiedzi. Mechanizm uwagi posiada więc cechy wspólne z siecią pamięci, ponieważ obie techniki pozwalają na dostęp do informacji z przeszłości. Ich działanie jest podobne, ponieważ w sieciach pamięci zakodowaną informację można postrzegać jako zakodowane zdanie źródłowe w mechanizmie uwagi [22]. Obecnie mechanizm uwagi jest wykorzystywany w wielu dziedzinach związanych z uczeniem maszynowym. W kontekście generowania dialogów może być używany do skupienia się na określonych słowach lub frazach w poprzednich wypowiedziach, aby wygenerować bardziej spójną i logiczną odpowiedź. Pozwala to na ograniczeniu zużycia mocy komputerowej na mniej istotnych częściach tekstu. Model transformer został przedstawiony na rys. 2.2 [21].



Rysunek 2.2: Model transformer [21]

Transformer to model architektury składający się z dwóch głównych komponentów: enkodera i dekodera. W oryginalnej pracy [21] każdy z tych komponentów jest zbudowany z sześciu identycznych warstw. Enkoder składa się z dwóch głównych podwarstw w każdej warstwie. Pierwsza z nich to *multi-head attention*, który jest rozwinięciem mechanizmu uwagi. W *multi-head attention* model oblicza wiele zestawów wag uwagi (tzw. "głów") dla każdego elementu danych wejściowych. Każda głowa ma swoje własne wagi, które są używane do obliczenia ważonych sum. Wyniki z każdej z głów są następnie łączone, aby uzyskać końcowy wynik. Drugą podwarstwą w enkoderze jest sieć *Feed Forward*, znana również jako sieć z propagacją w przód. Składa się z dwóch kroków: pierwszy to liniowa transformacja, która przekształca wektor wejściowy o wymiarze *embeddingu* (wynik z *multi-head attention*) na nowy wektor o wyższej wymiarowości, a następnie stosowana jest funkcja aktywacji, najczęściej ReLU (*ang. rectified linear unit*), która wprowadza nieliniowość do modelu. *Embedding* oznacza proces przekształcania wejściowych i wyjściowych tokenów na wektory o stałym wymiarze, zazwyczaj $d = 512$ w przypadku tego modelu. Przedstawia to sposób, w jaki tokeny są reprezentowane w przestrzeni wektorowej, co umożliwia ich dalsze przetwarzanie przez model. To pozwala modelowi na naukę bardziej skomplikowanych zależności między danymi. Kluczową zaletą tego podejścia jest to, że obliczenia dla każdego elementu w sekwencji są niezależne, co umożliwia równoległe przetwarzanie danych i znaczaco przyspiesza uczenie. Dekoder jest skonstruowany w sposób podobny do enkodera, również składając się z sześciu warstw. Jednakże, w dekoderze dodana jest trzecia podwarstwa znana jako *masked multi-head attention*. Jest to rozszerzenie standardowej warstwy *multi-head attention*, które wprowadza maskowanie. Maska jest stosowana, aby wykluczyć pewne informacje, które nie powinny

być dostępne w danym kontekście. Dzięki temu model może efektywnie przewidywać kolejne elementy sekwencji, korzystając wyłącznie z informacji dostępnych do danego momentu. *Positional encoding* polega na dodaniu dodatkowych wartości do wektorów wygenerowanych przez *embedding*, które reprezentują pozycje tokenów w sekwencji. Model transformer nie ma wbudowanej informacji o kolejności tokenów w sekwencji, dlatego *positional encoding* jest niezbędny, aby model mógł rozróżniać tokeny na podstawie ich pozycji w sekwencji. Dzięki temu nawet identyczne tokeny w różnych miejscach w sekwencji mają różne reprezentacje, co pozwala na efektywne uczenie się zależności między tokenami. Na końcu, funkcja *softmax* jest używana do przekształcania wyjścia dekodera na prawdopodobieństwa, które sumują się do 1. Te prawdopodobieństwa interpretowane są jako przewidywane prawdopodobieństwa dla następnego tokenu w sekwencji.

2.3.3. Duże modele językowe

Duże modele językowe LLM (*ang. large language models*) zazwyczaj odnoszą się do modeli językowych opartych na architekturze transformer, które zawierają setki miliardów (lub więcej) parametrów i następnie są trenowane na olbrzymich zbiorach danych tekstowych [23]. Modele LLM wykazują duże zdolności rozumienia języka naturalnego i rozwiązywania złożonych problemów poprzez generowanie tekstu, przez co znacząco wpłynęły na dziedzinę NLP w zakresie między innymi generowania dialogów i odpowiadania na pytania. Ważnym elementem ewolucji LLM jest prawo skalowania (*ang. scaling law*), które w kontekście LLM odnosi się do empirycznych obserwacji dotyczących zależności między rozmiarem modelu a jego zdolnościami i wydajnością. W pracy [24] wykazano, że większe modele LLM są w stanie uzyskać lepsze wyniki w zadaniach językowych, takich jak generowanie tekstów, tłumaczenie czy analiza semantyczna. Oznacza to, że im większa ilość danych i dłuższy czas trenowania tym lepsze możliwości modelu. W przypadku bardzo dużych LLM wykorzystuje się również różne techniki poprawy skuteczności jak *fine-tuning*, czyli dostosowanie wytrenowanego już modelu do konkretnego zadania poprzez dalsze trenowanie na mniejszym zbiorze danych związanych z danym zadaniem. To pozwala na dostosowanie modelu do specyficznych potrzeb aplikacyjnych, co jest szczególnie istotne w przypadku zastosowań takich jak tłumaczenie maszynowe, generowanie treści czy generowanie dialogów.

2.4. Zaawansowane techniki generowania dialogu dla rozwoju postaci NPC (Gafe)

fke)

Jak zauważono w poprzednim rozdziale, dialogi odgrywają kluczową rolę w budowaniu imprez i wpływają znacząco na emocjonalne zaangażowanie graczy w trakcie rozgrywki. Istotne jest, aby dialogi były dynamiczne i oryginalne, unikając powtarzalności i schematyczności, co mogłoby prowadzić do odczucia monotonii przez graczy podczas interakcji z postaciami niezależnymi. W odpowiedzi na te potrzeby, rozwój metod tworzenia dialogów, począwszy od tradycyjnych drzew dialogowych, aż po innowacyjne zastosowanie technologii sztucznej inteligencji, stał się przedmiotem dogłębnego analiz i badań naukowych. Ten rozdział skupia się na eksploracji najbardziej efektywnych i popularnych technik kreowania dialogów, a także omawia wyzwania i przyszłe ścieżki rozwoju tej dziedziny.

2.4.1. Drzewa dialogowe i interaktywna narracja

Drzewa dialogowe stanowią fundamentalną, a zarazem wyjątkowo skuteczną strategię w kreowaniu dialogów interaktywnych w świecie gier. Są one kluczem do konstruowania skomplikowanych opowieści, w których decyzje podejmowane przez graczy kierują rozwojem fabuły. Na poziomie technicznym, drzewa składają się z węzłów decyzyjnych, gdzie każdy z nich otwiera przed graczem nowe, potencjalnie odmienne konsekwencje w ramach gry.

Projekt "Façade" autorstwa Mateasa i Sternia [25] stanowi przykład zaawansowanego wykorzystania tej metody do stworzenia interaktywnego dramatu. W ich pracy drzewa dialogowe umożliwiają rozwijanie zaawansowanych interakcji pomiędzy postaciami a graczem. Ich badania podkreślają, jak dzięki drzewom dialogowym możliwe jest kształtowanie wyrazistych postaci oraz dynamicznych, reagujących na decyzje gracza narracji, oferujących mu różnorodne zakończenia. Ryan w artykule [26] rozwinał tę ideę, wskazując na możliwość stworzenia przez takie interaktywne narracje "awatarów historii" (*ang. avatars of story*). Pozwala to graczom na głębokie zanurzenie się w wątki fabularne, eksplorując je poprzez złożone interakcje dialogowe, co wprowadza nowy wymiar doświadczenia narracyjnego w grach.

2.4.2. Systemy punktacji i atrybutów

Systemy oceniania i atrybutów odgrywają kluczową rolę w kształtowaniu dialogów w grach, poprzez analizę określonych właściwości postaci kontrolowanej przez gracza. Dla przykładu, postaci o wyższym poziomie inteligencji mogą być przyznawane dodatkowe, bardziej złożone opcje dialogowe. Koncepcja wykorzystania atrybutów w mechanice gier (również systemach dialogowych) ma swoje korzenie w tradycyjnych grach fabularnych, takich jak Lochy i Smoki D&D (*ang. Dungeons & Dragons*), które pojawiły się w latach 70. XX wieku [27]. Z biegiem czasu systemy te przeszły ewolucję i zostały zaadaptowane do środowiska wirtualnego, zachowując przy tym swoją pierwotną rolę w kształtowaniu interakcji i rozwoju postaci.

W artykule [28] autor analizuje, jak systemy oceniania i atrybutów mogą wpływać na dialogi i rozwój postaci w grach. Systemy te funkcjonują poprzez przypisywanie wartości liczbowych do różnych cech postaci gracza, takich jak siła czy inteligencja. Te wartości są następnie brane pod uwagę podczas podejmowania decyzji dialogowych, co decyduje o dostępności określonych opcji dialogowych. Z kolei Ryan [26] podkreśla potrzebę zbalansowanego projektowania systemów ocen i atrybutów, aby zapewnić, że wybory dokonywane przez graczy są znaczące i mają realny wpływ na rozwój gry. Techniczna realizacja tego zadania wymaga ścisłej integracji systemu ocen z logiką gry i mechaniką dialogową, co jest osiągane poprzez wykorzystanie zaawansowanych struktur danych i algorytmów decyzyjnych.

2.4.3. Generowanie dialogów na podstawie AI

Wykorzystanie sztucznej inteligencji (AI) w generowaniu dialogów otwiera nowe perspektywy dla tworzenia interakcji, które są zarówno dynamiczne, jak i reaktywne. AI ma zdolność do analizowania kontekstu rozgrywki oraz wcześniejszych wyborów dokonanych przez gracza, co pozwala na tworzenie dialogów, które są nie tylko spójne, ale i nieoczywiste. Niemniej jednak, wdrażanie takich rozwiązań wymaga zastosowania zaawansowanych technik przetwarzania języka naturalnego oraz modelowania kontekstu.

Autorzy artykułu [29] badali potencjał sztucznej inteligencji w kreowaniu dialogów adaptacyjnych, które dostosowują się do decyzji i działań podejmowanych przez gracza. W praktyce, sys-

temy te mogą wykorzystywać metody uczenia maszynowego do analizy poprzednich decyzji gracza, generując odpowiedzi, które najlepiej odpowiadają narastającemu kontekstowi gry. To podejście wymaga nie tylko skomplikowanego przetwarzania języka naturalnego, ale również głębo-kiego modelowania kontekstu.

Short w swojej książce [30] dogłębnie opisuje proces wytwarzania gier i poszerza tę koncepcję, eksplorując ideę proceduralnego generowania samej gry w tym dialogów, co przyczynia się do tego, że dialogi są kreowane w czasie rzeczywistym. Realizacja tego założenia wymaga wykorzystania algorytmów generatywnych, zdolnych do tworzenia koherentnych i zaangażowanych linii dialogowych, bazujących na zdefiniowanych parametrach narracyjnych oraz kontekście gry.

2.4.4. Lokalizacja i adaptacja kulturowa

Równie ważnym z punktu widzenia tworzenia dialogów jest lokalizacja i adaptacja kulturowa, by umożliwić globalny dostęp i zapewnić, że gra trafia do odbiorców na całym świecie. W [31] autor akcentuje istotność lokalizacji dialogów, podkreślając, że zadanie to wymaga nie tylko przekładu tekstopiowego, ale także kulturowej adaptacji treści dialogowej, aby była ona odpowiednia dla różnych odbiorców. Proces ten wymaga dogłębnej analizy kulturowej, zrozumienia kontekstu, a także technicznej adaptacyjności systemów dialogowych, co pozwala na łatwe wprowadzanie zmian i dostosowań.

2.4.5. Wyzwania i ograniczenia

Rozwój dialogów w grach wiąże się z licznymi wyzwaniami, takimi jak zarządzanie rozbudowanymi drzewami dialogowymi, włączanie systemów punktacji i atrybutów, a także opracowywanie skomplikowanych modeli AI do kreowania dialogów. Kompleksowe drzewa dialogowe mogą okazać się wyzwaniem w obszarze zarządzania i debugowania, szczególnie w obszernych projektach oferujących liczne narracyjne ścieżki wyboru. Ponadto, osiągnięcie naturalności i spójności dialogów generowanych przez AI wymaga nieustannego udoskonalania modeli językowych oraz ich weryfikacji przez użytkowników. Ograniczenia w narracji mogą pojawić się także w próbach utrzymania spójności opowieści przy zapewnieniu graczom możliwości wpływania na jej przebieg. Balansowanie między wolnością gracza a narracyjną spójnością stanowi znaczące wyzwanie. Dodatkowo, lokalizacja dialogów nie ogranicza się wyłącznie do tłumaczenia, ale wymaga także kulturowej adaptacji, co może wprowadzać kolejne ograniczenia dla treści dialogowych.

2.4.6. Przyszłe kierunki rozwoju

Rozwój w obszarze sztucznej inteligencji i przetwarzania języka naturalnego nadal otwiera nowe możliwości dla tworzenia dynamicznych i reaktywnych dialogów. Modele AI mają potencjał do stania się bardziej zaawansowanymi w rozumieniu kontekstu rozgrywki i preferencji graczy, co umożliwia kształtowanie bardziej spersonalizowanych i wciągających interakcji. Przyszłe technologie mogą pozwolić na stworzenie jeszcze bardziej interaktywnych i adaptacyjnych opowieści, w których decyzje graczy wywierają silniejszy wpływ na kształtowanie historii i świata gry. Możliwe jest również pojawienie się adaptacyjnych dialogów, ewoluujących w czasie rzeczywistym w reakcji na działania gracza, stan świata gry, a nawet emocjonalny stan postaci.

2.4.7. Podsumowanie

Wykorzystanie drzew dialogowych i zaawansowanych modeli AI umożliwia tworzenie złożonych, dynamicznych interakcji, które są spersonalizowane i responsywne wobec działań gracza. Jednocześnie, systemy punktacji i atrybutów dodają kolejną warstwę głębi, pozwalając na rozbudowę charakterystyki postaci i wpływanie na przebieg narracji. Lokalizacja i adaptacja kulturowa rozszerzają zasięg gier, czyniąc je dostępnymi i rezonującymi z globalną publicznością. Wyzwania, takie jak utrzymanie spójności narracyjnej, zarządzanie złożonością dialogów oraz kulturowa adaptacja, pozostają istotne. Przyszłość zapowiada się obiecująco, z potencjałem do dalszego zwiększania immersji i interaktywności w grach dzięki nowym technologiom i metodologią.

2.5. Metody oceny jakości generowanych dialogów (Gaffke)

Aby zweryfikować, czy dialogi generowane przez NPC odpowiadają założeniom omówionym we wstępie, należy skonstruować metody pomiaru jakości. W literaturze istnieje wiele podejść do tego zagadnienia. W niniejszym rozdziale skupimy się na wybranych z nich. Testy oceniające jakość generowanego tekstu w dziedzinie NLP można podzielić na kilka kategorii.

2.5.1. Ewaluacja automatyczna

Pierwszą z nich jest ewaluacja automatyczna, polegająca na użyciu automatycznych metryk oceny. Ich celem jest sprawdzenie, w jakim stopniu słowa w zdaniach wygenerowanych przez chatboty pokrywają się ze zdaniem napisanym przez ludzi. Wśród standardów tej oceny można wyróżnić BLEU [32], analizujący współwystępowanie n-gramów w rzeczywistych odpowiedziach (*ang. ground truth*) i proponowanych odpowiedziach. Standard BLEU oblicza precyzę n-gramów dla całego zbioru danych:

$$P_n(r, \hat{r}) = \frac{\sum_k \min(h(k, r), h(k, \hat{r}_i))}{\sum_k h(k, r_i)} \quad (2-1)$$

gdzie,

- k oznacza indeksowanie wszystkich możliwych n-gramów o długości n ,
- $h(k, r)$ to liczba wystąpień n-gramów k w odpowiedzi r .

Wartość wyrażenia mieści się w zakresie od 0 do 1. Z uwagi na faworyzowanie krótszych zdań, wprowadzono modyfikację tej metody - BLEU-N, z dodatkową karą za zbyt krótkie zdanie. Standard ten z racji na swoją bezpośrednią w sprawdzaniu występowania słów, został wykorzystany głównie do sprawdzania tłumaczeń. Rozwinięciem metody BLEU jest METEOR [33]. Pomaga on rozwiązać problemy wynikające ze zbyt dosłownego traktowania słów i porównywania ich 1:1. W metodzie tej uwzględnione zostały również synonimy, wyrazówki, a także synonimy słów z bazy WordNet. Porównanie odbywa się zgodnie ze standardem BLEU.

Przedstawione powyżej metryki należą do grupy tak zwanego podobieństwa nakładania się słów (*ang. word-overlap similarity*). Inną grupą są metryki oparte na osadzaniu (*ang. embedding-based metrics*), które porównują znaczenie całego zdania. Aby to osiągnąć, tworzy się numeryczne odzwierciedlenie zdania, przekształcając poszczególne słowa w wektory osadzenia. Przykładem takiej techniki jest Word2Vec [34], która korzysta z semantyki dystrybucyjnej, analizując współwystępowanie słów w tekście. Wektory są następnie porównywane przy użyciu miary odległości kosinusowej:

$$EA := \cos(\bar{e}_r, \bar{e}_{\hat{r}}) \quad (2-2)$$

Miara ta porównuje kierunki i długości tych wektorów w przestrzeni. Przykładem takiej metryki jest średnie osadzenie (*ang. embedding average*), dla której wartość wektora osadzenia zdania oblicza się na podstawie średniej osadzeń pojedynczych słów:

$$\bar{e}_r = \frac{\sum_{w \in r} e_w}{|\sum_{w' \in r} e_{w'}|} \quad (2-3)$$

gdzie,

– e jest wartością osadzenia słowa r .

Oprócz wymienionych metod istnieje wiele różnych pochodnych standardów oceny automatycznej. Jak jednak wskazano w artykule [35], w którym zostało przeprowadzone badanie sprawdzające oba typy metod oceny automatycznej na danych technicznych i nietechnicznych. Metody te bez nadzoru nie korelują z ocenami ludzi. Autorzy w swoich wnioskach wspominają że metryki te nie są wystarczające i mogą służyć jedynie jako narzędzie pomocnicze dla oceny ludzkiej.

2.5.2. Ewaluacja heurystyczna

Ocena systemu dialogowego wymaga sprawdzenia, czy wygenerowane odpowiedzi są zgodne z kontekstem i fabułą. W tym celu opracowuje się specjalne scenariusze, które mają skłonić system do udzielenia konkretnej odpowiedzi. Jak zauważają autorzy artykułu [36], wskaźniki oceny mogą mieć charakter subiektywny lub obiektywny. W podejściu subiektywnym, użytkownik formułuje pytania zgodnie z zaplanowanym scenariuszem, aby uzyskać odpowiedź na dany temat. Przykładem z artykułu jest uzyskanie informacji o restauracji. Użytkownik na końcu dialogu ocenia, czy otrzymana odpowiedź jest zgodna z oczekiwaniemi. Ta metoda zależy od indywidualnego postrzegania użytkownika i może prowadzić do różnych ocen tej samej odpowiedzi. W odpowiedzi na ograniczenia podejścia subiektywnego, rozwinięto metody oceny obiektywnej. Wprowadza się w nich współczynnik celu, który określa, w jakim stopniu informacje dostarczone przez system są zgodne z oczekiwaniemi. Autorzy wspominają również o zastosowaniu algorytmów heurystycznych, które analizują logi odpowiedzi i przydzielają punktację na podstawie ich adekwatności w odniesieniu do pytania. W obiektywnym podejściu rozróżnia się dwa rodzaje wskaźników sukcesu: jeden oparty na celach użytkownika wyznaczonych w scenariuszu, a drugi na celach wnioskowanych z pytań użytkownika. Pierwszy zakłada, że użytkownik ściśle przestrzega scenariusza, choć często zdarza się, że użytkownik odbiega od niego, co może prowadzić do negatywnej oceny. Drugi model polega na analizie przez algorytm kontekstu zadanych pytań, co pozwala na bardziej precyzyjną ocenę zgodności odpowiedzi z pytaniem. Artykuł [36] wskazuje, że oceny subiektywne są zazwyczaj wyższe niż oceny obiektywne, co autorzy tłumaczą rygorystyczną naturą używanych algorytmów heurystycznych.

Na szczególne wyróżnienie zasługują metody heurystyczne dla systemów nienastawionych na cel. Przykładem takiego systemu może być zyskujący ogólną popularność w ostatnich latach ChatGPT, który nie posiada ściśle określonego zakresu czy tematyki [37]. Do tego rodzaju systemów wykorzystywany był znany już od wielu lat, a opracowany w latach 50. ubiegłego wieku test Turinga [38]. Polega on na wygenerowaniu odpowiedzi przez system dialogowy i zmieszaniu ich z odpowiedziami napisanymi przez ludzi. Arbitrzy, którymi są inne osoby miały ocenić czy dana odpowiedź została wygenerowana przez system LLM czy napisana przez człowieka. Testy te jednak są już dzisiaj nieaktualne. Jak przeczytać możemy w artykule [39] już w 2005 został uznany za niewystarczający, a w dobie dzisiejszych systemów, chociażby wspomnianym wcześniej GPT

(ang. *generative pre-trained transformer*), jest całkowicie bezużyteczny.

2.5.3. Ocena użytkowników

W niniejszym fragmencie przechodzimy do istotnego aspektu naszej pracy, jakim jest ocena systemu dialogowego przez użytkowników. W tym kontekście koncentrujemy się na aspektach najbardziej istotnych dla naszego badania. W wielu pracach naukowych dotyczących oceny systemów dialogowych wykorzystywanych przez postacie niezależne (NPC) w grach, takich jak te opisane w publikacjach [40] i [41], analiza opiera się na ocenie użytkowników, co jest szczególnie ważne przy wykorzystaniu wcześniej wytrenowanych modeli językowych (LLM). Autorzy tych artykułów stosują ankiety mające na celu ocenę wrażeń użytkowników, ale także spójności kontekstowej i innych aspektów. Wśród powtarzających się kryteriów oceny znajdują się: płynność wypowiedzi, kontekstowość, kreatywność, wiarygodność, stopień powtórzeń, oddanie roli i klimatu rozgrywki. Większość tych cech jest trudna do oceny za pomocą testów automatycznych i zależy przede wszystkim od opinii graczy.

2.5.4. Samoocena

Jest to stosunkowo nowa metoda oceny systemów dialogowych, która pojawiła się wraz z dynamicznym rozwojem systemów opartych na dużych modelach językowych LLM *large language models*. Ta metoda, znana jako samoocena, może być realizowana przez ten sam model, który generuje odpowiedzi, lub przez inny, niezależny model. Polega ona na zadawaniu pytania modelowi o to, jak ocenia swoją własną wypowiedź. Zgodnie z informacjami z artykułu [42], metoda ta, oparta na modelu GPT-3, potrafi wykrywać nieodpowiednie lub wulgarne treści oraz oceniać dialog pod względem jego zgodności z rodzajem i gatunkiem gry. Autorzy zauważają, że przy prostych pytaniach nie jest konieczne podawanie szerokiego kontekstu. Jednakże, dostarczenie przykładów i wskazówek dotyczących optymalnych wypowiedzi znaczaco zwiększa skuteczność oceny. Zastosowanie dodatkowego uszczegółowienia pytania i żądanie odpowiedzi numerycznej, zamiast wolnej odpowiedzi w formie zdania, pozwoliło znacznie zwiększyć zgodność ocen z ocenami użytkowników.

2.5.5. Podsumowanie

Podsumowując, ocena jakości i naturalności dialogów generowanych przez metody sztucznej inteligencji może być przeprowadzana za pomocą różnorodnych technik. Wśród metod automatycznych wyróżniamy metryki takie jak BLEU, METEOR oraz oparte na osadzeniach wektorowych, takie jak *Embedding Average*, które analizują zgodność i semantykę generowanych odpowiedzi w porównaniu do odpowiedzi ludzkich. Te metody, choć użyteczne, często wymagają nadzoru ludzkiego, gdyż nie zawsze korelują z ludzkimi ocenami jakości dialogu. Weryfikacja heurystyczna zapewnia bardziej kontekstową analizę, biorąc pod uwagę spójność odpowiedzi z fabułą i scenariuszem, oraz wykorzystując algorytmy heurystyczne do analizy logów odpowiedzi i przypisywania im punktacji. Dwa główne podejścia to obiektywny wskaźnik sukcesu na podstawie przydzielonych celów użytkownika oraz wnioskowanych celów, co pozwala na bardziej elastyczną ocenę systemu dialogowego. Ocena użytkowników jako najbardziej bezpośrednia metoda, pozwala na ocenę płynności, kontekstowości, kreatywności, wiarygodności i innych subiektywnie odbieranych aspektów dialogu. Ankiety i bezpośrednie opinie graczy są tutaj kluczowe, ponieważ wiele cech dialogów generowanych przez AI nie jest łatwe do zmierzenia za pomocą automatycznych metryk.

Nowością w ocenie systemów dialogowych jest samoocena oparta na modelach językowych takich jak GPT, które są w stanie ocenić swoje własne odpowiedzi pod kątem niepożądanych treści lub zgodności z kontekstem gry. Choć ten rodzaj oceny wciąż się rozwija, prezentuje obiecujące możliwości dla przyszłej automatyzacji procesów oceny. W kontekście gier, gdzie interakcje z NPC są kluczowe dla immersji i doświadczenia gracza, ocena dialogów systemów AI odgrywa szczególnie ważną rolę. Dostosowanie metod oceny do specyfiki gier, w tym fabuły, scenariusza i interakcji z graczem, jest zatem niezbędne do stworzenia przekonujących i zaawansowanych NPC.

3. TECHNOLOGIE, ALGORYTMY I NARZĘDZIA

Ten rozdział stanowi przegląd technologii, algorytmów i narzędzi wykorzystanych w procesie tworzenia prototypowej gry. Te informacje mają na celu wgląd w techniczne podwaliny, na których opiera się projekt. W kolejnych sekcjach omówione zostaną wszystkie wykorzystane komponenty technologiczne, w tym duży model językowy wraz z jego interfejsem programistycznym, silnik Unity, technologia chmury, na której zapisywane są wszystkie dane oraz wykorzystane środowisko programistyczne i język programowania.

3.1. GPT (Lange)

GPT, czyli Generative Pre-trained Transformer, jest modelem językowym opartym na architekturze Transformer, który został stworzony przez OpenAI. Został on zaprojektowany do generowania tekstu na podstawie dostarczonych mu danych wejściowych. Najnowszą iteracją jest GPT-4o, która wyróżnia się zwiększoną skalą, lepszą jakością generowanego tekstu oraz szerszym zakresem zastosowań w porównaniu do poprzednich iteracji [43]. U podstaw GPT leży architektura transformer opisana w 2.3.2. Mechanizm uwagi, który wykorzystywany jest w transformerze umożliwia modelowi skuteczniejsze wychwytywanie zależności w sekwencjach tekstu znacznie przewyższając w wynikach poprzednie podejścia takie jak RNN czy LSTM.

3.1.1. Proces uczenia GPT

Model jest trenowany na ogromnych zbiorach danych tekstowych, dzięki czemu uczy się on statystycznych zależności między słowami i frazami, co pozwala na generowanie tekstu podobnego do ludzkiego. Istotną cechą jest również zdolność do generowania tekstu w sposób bezwzględny, czyli niewymagający żadnego kontekstu ani podpowiedzi do rozpoczęcia generowania. GPT jest trenowany w dwóch głównych etapach [44]:

- Wstępne uczenie nienadzorowane (*ang. unsupervised pre-training*)
- Nadzorowane dostrajanie (*ang. supervised fine-tuning*)

We wstępny uczeniu nienadzorowanym model jest trenowany na dużym zbiorze tekstów, który obejmuje różnorodne źródła, takie jak książki, artykuły i strony internetowe. Celem uczenia nienadzorowanego jest nauczenie modelu zrozumienia ogólnych wzorców językowych, takich jak gramatyka, składnia i semantyka. Proces ten polega na prognozowaniu następnego słowa w sekwencji na podstawie poprzednich słów, co pozwala modelowi uczyć się kontekstu i struktury języka na szeroką skalę. W kolejnym etapie model jest dalej trenowany na bardziej specyficznych zestawach danych, które są dostosowane do konkretnych zastosowań. Fine-tuning pozwala modelowi dostosować swoje umiejętności do bardziej wyspecjalizowanych zadań. W tej fazie wykorzystuje się technikę nadzorowanego uczenia, gdzie model jest dostosowywany na podstawie przykładów odpowiednich interakcji [44].

3.1.2. GPT-4

GPT-4 jest czwartą iteracją modelu GPT i bezpośrednim następcą GPT-3. Nowy model znacznie zwiększył liczbę parametrów i jest ich kilkakrotnie więcej niż w poprzednim modelu, który

posiadał ich 175 miliardów (dokładna liczba parametrów GPT-4 nie została ujawniona) [37]. Takie zwiększenie liczby parametrów znacząco poprawiło wychwytywanie bardziej złożonych wzorców językowych, dzięki czemu wykazał doskonałą wydajność w różnych zadaniach rozumienia i generowania tekstu w porównaniu z poprzednikami. Zgodnie z [43] w testach rozumienia tekstu wykazał się 80.9% dokładnością, podczas gdy poprzedni model miał wynik 64.1%. Podobnie w testach rozumowania, testach wielokrotnego wyboru, czy zadaniach matematycznych model 4 zawsze osiągał wynik lepszy o co najmniej 10 punktów procentowych. Z tego samego powodu podjęto decyzję o stworzeniu gry całkowicie w języku angielskim, ponieważ model we wszystkich testach wykazał przewagę w generowanych odpowiedziach właśnie w tym języku, co daje większą szansę na lepszy rezultat w przypadku prowadzonych dialogów z graczami.

3.1.3. *Interfejs programistyczny OpenAI*

Interfejs programistyczny API (*ang. application programming interface*) udostępniane przez OpenAI, umożliwia programistom łatwy dostęp do zaawansowanych funkcji modeli GPT, takich jak generowanie tekstu, analizy językowej i tłumaczenia. API jest zaprojektowane w sposób, który ułatwia integrację z różnymi aplikacjami, oferując szeroki zakres możliwości, które można dostosować do specyficznych potrzeb użytkowników. Struktura API opiera się na modelu zapytań i odpowiedzi. Użytkownicy wysyłają zapytania do serwera API, który przetwarza je za pomocą modelu GPT i zwraca wygenerowane odpowiedzi. Kluczowym elementem API są endpointy, które obsługują różne funkcje modelu. Endpointy te są zoptymalizowane pod kątem konkretnych zadań, takich jak generowanie tekstu, klasyfikacja tekstu, ekstrakcja informacji, parafrasowanie tekstu, streszczenie oraz tłumaczenie językowe, co pozwala na efektywne wykorzystanie zasobów modelu.

Dostęp do API jest chroniony za pomocą kluczy API, które są generowane dla każdego użytkownika. Klucze można uzyskać po zalogowaniu się na platformie OpenAI poprzez zakładkę API w panelu nawigacyjnym. Muszą być dołączane do każdego zapytania w celu uwierzytelnienia użytkownika i monitorowania zużycia zasobów. Każde zapytanie do API składa się z kilku kluczowych komponentów: tekstu wejściowego oraz opcjonalnych parametrów, które kontrolują zachowanie modelu, takich jak długość wygenerowanego tekstu, stopień kreatywności (*temperature*), liczba prób (*n*), maksymalna liczba tokenów, stopień penalizacji powtarzających się fraz oraz wersja modelu. Odpowiedzi API są zwracane w formacie JSON, który zawiera wygenerowany tekst oraz dodatkowe metadane, takie jak czas przetwarzania zapytania i użyte parametry. Odpowiedzi te mogą być następnie przetwarzane przez aplikacje klienckie zgodnie z ich specyficznymi potrzebami.

Proces przetwarzania zapytania w API zaczyna się od wysłania zapytania HTTP POST na odpowiedni punkt dostępu (*ang. endpoint*) API, z dołączonym kluczem API i danymi wejściowymi. Serwer API odbiera zapytanie, weryfikuje klucz API oraz sprawdza poprawność danych wejściowych i parametrów. Po weryfikacji, serwer przekazuje dane wejściowe do modelu GPT, który analizuje tekst, uwzględniając dostarczone parametry. Model GPT generuje odpowiedź na podstawie analizowanego tekstu wejściowego, przewidując kolejne słowa w sekwencji zgodnie z wyuczonymi wzorcami językowymi. Wygenerowana odpowiedź jest następnie formułowana w formacie JSON i zwracana do użytkownika.

```
POST /v1/engines/chat/completions HTTP/1.1
Host: api.openai.com
Authorization: Bearer YOUR_API_KEY
```

```

Content-Type: application/json

{
  "model": "gpt-3.5-turbo",
  "messages": [
    {"role": "system", "content": "You are an NPC in a fantasy game."},
    {"role": "user", "content": "What is your name?"}
  ],
  "max_tokens": 150,
  "temperature": 0.7,
  "top_p": 1.0,
  "frequency_penalty": 0,
  "presence_penalty": 0
}

```

Przykład zapytania do API, gdzie:

- *POST /v1/engines/chat/completions HTTP/1.1*: Określa metodę HTTP (POST) i endpoint (/v1/engines/chat/completions), który obsługuje żądanie generowania tekstu.
- *Host: api.openai.com*: Adres serwera API OpenAI.
- *Authorization: Bearer YOUR_API_KEY*: Nagłówek autoryzacji zawierający klucz API, który uwierzytelnia użytkownika.
- *Content-Type: application/json*: Określa, że treść zapytania jest w formacie JSON.
- *model*: Wybór modelu, w tym przypadku "gpt-3.5-turbo".
- *messages*: Lista wiadomości, gdzie każda wiadomość jest obiektem zawierającym klucz "role" oraz "content".
 - *system*: Określa kontekst, ustawienia lub rolę, którą model ma przyjąć.
 - *user*: Wiadomość od użytkownika, na którą model powinien odpowiedzieć.
 - *assistant*: Poprzednie odpowiedzi modelu w kontekście rozmowy.
- *content*: Tekst wiadomości.
- *max_tokens*: Maksymalna liczba tokenów w odpowiedzi.
- *temperature*: Parametr kontrolujący kreatywność odpowiedzi. Zakres od 0.0 do 1.0, gdzie 1.0 oznacza brak ograniczeń, niższe wartości oznaczają bardziej skoncentrowane wybory.
- *top_p*: Parametr dla próbkowania top-p zwanego także próbkowaniem jądra (*ang. nucleus sampling*). Zakres od 0.0 do 1.0, gdzie 1.0 oznacza brak ograniczeń, niższe wartości oznaczają bardziej skoncentrowane wybory. Alternatywa dla parametru temperature.
- *frequency_penalty*: Parametr od -2.0 do 2.0, który zmniejsza prawdopodobieństwo powtarzania tych samych linii. Wartości dodatnie zmniejszają częstotliwość powtarzania.
- *presence_penalty*: Parametr od -2.0 do 2.0, który zwiększa prawdopodobieństwo mówienia o nowych tematach. Wartości dodatnie sprawiają, że model częściej wprowadza nowe tematy.

Dla tak skonstruowanego zapytania, odpowiedź od serwera może wyglądać następująco:

```
{
  "id": "chatcmpl-1234abcd",
  "object": "chat.completion",
  "created": 1677779112,
  "model": "gpt-3.5-turbo",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Odpowiedź"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 30,
    "completion_tokens": 20,
    "total_tokens": 50
  }
}
```

- *id*: Unikalny identyfikator odpowiedzi
- *object*: Typ obiektu odpowiedzi.
- *created*: Znacznik czasu, kiedy odpowiedź została wygenerowana.
- *model*: Model użyty do wygenerowania odpowiedzi, w tym przypadku "gpt-3.5-turbo".
- *model*: Wybór modelu, w tym przypadku "gpt-3.5-turbo".
- *choices*: Lista możliwych odpowiedzi wygenerowanych przez model.
 - *index*: Indeks odpowiedzi w liście możliwych odpowiedzi.
 - *message*: Treść wygenerowanej wiadomości.
 - * *role*: rola, identyczne znaczenie, jak w przypadku zapytania
 - * *content*: Wygenerowana treść odpowiedzi.
 - *finish_reason*: Powód zakończenia generowania odpowiedzi, w tym przypadku "stop".
- *usage*: Informacje o zużyciu tokenów podczas generowania odpowiedzi.
 - *prompt_tokens*: Liczba tokenów użytych w zapytaniu.
 - *completion_tokens*: Liczba tokenów wygenerowanych w odpowiedzi.
 - *total_tokens*: Łączna liczba tokenów użytych w żądaniu.

3.2. Unity (Lange)

Unity to zaawansowany silnik do tworzenia gier i aplikacji 2D, 3D, rzeczywistości rozszerzonej AR (*ang. augmented reality*) i rzeczywistości wirtualnej VR (*ang. virtual reality*). Narzędzie zostało stworzone przez Unity Technologies i pierwszy raz zaprezentowane w 2005 roku. Obecnie jest to jedno z najpopularniejszych rozwiązań do tworzenia gier wspierające różnorodne platformy, takie jak Windows, Linux, konsole, przeglądarki internetowe oraz urządzenia Apple.

3.2.1. Architektura Unity

Unity wykorzystuje architekturę opartą o komponenty. Oznacza to, że każdy obiekt w grze (*ang. GameObject*) może mieć przypisane różnorodne komponenty determinujące jego właściwości i zachowanie. Jest to, więc struktura modularna pozwalająca na łatwe dodawanie, usuwanie i modyfikowanie funkcjonalności obiektów. Kluczowe elementy architektury silnika to między innymi:

- *GameObject* - podstawowy element, do którego przypisuje się komponenty. Jest to dowolny obiekt w scenie gry, na przykład postać gracza, kamera, światło czy elementy środowiska.
- *Komponent* - jednostka funkcjonalna, którą dodaje się do *GameObject*. Każdy komponent realizuje specyficzną funkcjonalność, na przykład renderowanie grafiki, obsługę fizyki, czy kontrolę animacji.
- *Scena* - jest to podstawowa jednostka organizacyjna, która zawiera zbiór *GameObject*. Może reprezentować poziom gry, menu lub dowolny inny stan gry.
- *Skrypty* - napisane w języku C# odpowiadają za kontrolę interakcji, logiki gry i zachowania obiektów. De facto są one rozpoznawane jako komponenty, a więc w ten sam sposób mogą być dodawane do *GameObject*.

3.2.2. Edytor i narzędzia

Głównym narzędziem dewelopera, w którym tworzy, testuje i modyfikuje się projekty jest edytor Unity. Jest to interfejs graficzny umożliwiający zarządzanie scenami, obiektemi, komponentami i skryptami. W ramach tego ekosystemu możemy wyróżnić kilka kluczowych funkcjonalności. Głównym źródłem pozyskiwania zasobów i narzędzi do tworzenia gier jest sklep internetowy Unity Asset Store, widok sceny i widok gry zapewniają niezbędne perspektywy umożliwiające tworzenie i wizualizacje grafiki trójwymiarowej, a poszczególne okna ułatwiają zarządzanie zasobami, strukturą projektu i testowanie. W tym podrozdziale opisane zostaną najważniejsze aspekty edytora.

- Okno hierarchii (*hierarchy*) - umożliwia zobaczenie wszystkich obiektów gry obecnych na scenie. Kompozycja sceny ma charakter hierarchiczny w relacji rodzic-dziecko. Jest to narzędzie do zarządzania strukturą sceny i organizacji elementów gry.
- Okno inspektora (*inspector*) - zawiera szczegółowe informacje o obecnie wybranym obiekcie ze sceny lub zasobie. Umożliwia deweloperowi szybką modyfikację właściwości obiektu, dodawanie do niego komponentów bezpośrednio z poziomu interfejsu graficznego.
- Okno projektu (*project*) - wyświetla strukturę wszystkich zasobów w grze uporządkowanych na zasadzie folderów, jest to więc katalog dostępu do tekstur, skryptów, plików audio i innych plików.
- Konsola (*console*) - wyświetla ostrzeżenia, błędy i logi, służy jako narzędzie do debugowania.

Widok sceny

Jest to główny obszar roboczy służący do tworzenia i manipulowania środowiskiem 3D, czyli obiektami tworzącymi grę. Udostępnia szereg narzędzi pozwalających deweloperom na umieszczenie, skalowanie, obracanie obiektów oraz dostosowywanie oświetlenia, kamer i innych ustawień środowiska. Narzędzie to jest niezbędne do projektowania poziomów, rozmieszczania zasobów i projektowania kompozycji wizualnej gry. Nawigacja odbywa się za pomocą myszy i klawiatury, a

także skrótów klawiszowych i pozwala na dowolne poruszanie się w przestrzeni 3D w tym przesuwanie, powiększanie i obracanie kamery.

Widok gry

Widok gry to podgląd w czasie rzeczywistym tego, jak gra wygląda i zachowuje się w czasie prawdziwej rozgrywki. Odzwierciedla obraz z kamer umieszczonych na scenie i symuluje rzeczywistą rozgrywkę, w tym animacje, interakcje użytkownika oraz fizykę. Jest to niezbędne narzędzie do testowania gry poprzez umożliwienie obserwacji wszystkich zachowań i zmian w środowisku. Widok ten zawiera także narzędzia do debugowania i profilowania Unity poprzez obserwację wskaźników wydajności oraz rejestrację danych wyjściowych.

Unity Asset Store

Unity Asset Store to zintegrowana z edytorem platforma internetowa oferująca ogromną bibliotekę zasobów, narzędzi i usług dostępnych za darmo lub płatnych. Asset Store zapewnia szeroki zakres zasobów, w tym modele 3D, tekstury, animacje, efekty dźwiękowe i skrypty, które można wykorzystać do przyspieszenia rozwoju i poprawy jakości gier. Kluczowe kategorie to:

Modele 3D - sklep oferuje szeroki wybór darmowych modeli 3D począwszy od modeli środowiska i skończywszy na postaciach. Modele te zawierają gotowe tekstury, materiały i shadery.

Audio - duża gama plików audio, w tym efekty dźwiękowe, dźwięki otoczenia i całe ścieżki dźwiękowe.

Skrypty i wtyczki - gotowe skrypty dodające funkcjonalności do projekty gry lub też do edytora Unity. Są to na przykład gotowe systemy generowania proceduralnego, ruchu kamery czy zarządzania muzyką w grze.

Jest to narzędzie niezwykle przydatne i przede wszystkim pozwalające zaoszczędzić czas i wysiłek, tym samym umożliwiając skupienie się na podstawowej mechanice rozgrywki.

3.2.3. Unity Cloud

Unity Cloud jest platformą oferowaną przez Unity Technologies, która łączy w sobie wiele narzędzi i usług chmurowych zaprojektowanych w celu wsparcia tworzenia, wdrażania i zarządzania zawartością gry w czasie rzeczywistym. Jest to swoiste rozszerzenie silnika Unity asystujące w procesie rozwoju gier w każdym etapie cyklu rozwoju. Narzędzie jest zintegrowane bezpośrednio z edytorem Unity, a konkretne usługi udostępniane są poprzez komponenty Unity Cloud zawierające interfejsy API chmury Unity i SDK dające dostęp do zasobów i zarządzania nimi. Poniżej przedstawione są kluczowe usługi oferowane przez to narzędzie, które są szczególnie istotne w tworzeniu projektu do niniejszej pracy.

Collaborate

Jest to oparte na chmurze narzędzie systemu kontroli wersji i współpracy w projektach Unity. Umożliwia wielu członkom zespołu projektowego pracę nad tym samym projektem, a także śledzenie zmian i synchronizowanie w czasie rzeczywistym. Jest to również znaczne uproszczenie zarządzaniem zasobami i innymi plikami projektu, w szczególności płatnymi zasobami ze sklepu Asset Store obłożonymi licencjami.

Cloud Save

Usługa zapewniająca mechanizmy przechowywania i pobierania danych gracza. Dane są przechowywane najczęściej w formacie JSON i mogą obejmować dowolne wartości, takie jak statystyki, postępy w grze, ustawienia i wszelkie inne interakcje. Usługa pozwala także na odczyt danych umożliwiając tym samym zdalny odczyt postępów gracza i przywrócenie ich w momencie uruchomienia gry. Synchronizacja danych może być kontrolowana zarówno ręcznie jak i automatycznie co daje deweloperowi całkowitą kontrolę.

Player Management

Jest to system zarządzania profilami graczy, a także ich danymi i interakcjami. Umożliwia uwierzytelnianie i śledzenie aktywności graczy z poziomu aplikacji internetowej. Aby móc wykorzystać ten system należy uwierzytelić gracza, mogą to być zarówno metody klasyczne takie jak hasło, loginy społecznościowe lub też konta gości pozwalające na zachowanie pełnej anonimowości przy zachowaniu wszystkich istotnych funkcji. Po uwierzytelnieniu anonimowym każdy gracz jest zapisany przy pomocy automatycznie wygenerowanego UUID, który zapisywany jest w usłudze jak i lokalnie na jego komputerze.

3.3. Język programowania i środowisko programistyczne (Gaffke)

3.3.1. Język C#

C# to język programowania, który pełni kluczową rolę w realizacji projektu w Unity i jest natywnym rozwiązaniem [45]. Wybrany ze względu na swoją integrację z silnikiem Unity, C# umożliwia tworzenie logiki gry, zarządzanie interakcjami użytkownika oraz implementację złożonych mechanizmów. Jego obiektowe podejście do programowania pozwala na strukturalne organizowanie kodu, co ułatwia zarówno zarządzanie projektem, jak i późniejsze jego modyfikacje. W kontekście naszego projektu, C# umożliwia implementację systemów takich jak sztuczna inteligencja dla postaci niezależnych, systemy zarządzania stanem gry czy obsługa interfejsu użytkownika. Język ten, poprzez swoje silne typowanie, minimalizuje ryzyko błędów w kodzie, co jest szczególnie istotne w projektach o dużej złożoności. Dodatkowo, C# wspiera programowanie asynchroniczne, co pozwala na efektywne zarządzanie operacjami wymagającymi intensywnego przetwarzania, bez wpływu na płynność działania aplikacji. W naszym projekcie, asynchroniczne przetwarzanie jest kluczowe, na przykład w przypadku ładowania danych w tle lub obsługi sieci.

3.3.2. Środowisko Visual Studio

Visual Studio to zintegrowane środowisko programistyczne IDE (*ang. integrated development environment*), które zostało wybrane do tworzenia i zarządzania kodem C# w projekcie. Visual Studio oferuje zaawansowane narzędzia, które znaczco usprawniają pracę programisty, szczególnie w kontekście rozwoju oprogramowania w Unity. Jednym z kluczowych elementów, które przyczyniły się do wyboru Visual Studio, jest funkcja IntelliSense. Narzędzie to wspomaga pisanie kodu poprzez automatyczne podpowiedzi i korekty, co minimalizuje błędy oraz przyspiesza proces programowania. W naszym projekcie, IntelliSense ułatwia pracę nad złożonymi algorytmami i integracją systemów. Kolejnym istotnym aspektem jest debugger, który umożliwia analizowanie i testowanie kodu w czasie rzeczywistym. Dzięki temu możliwe jest szybkie wykrywanie i naprawia-

nie błędów, co jest kluczowe dla utrzymania wysokiej jakości oprogramowania. Visual Studio jest również w pełni zintegrowane z Unity, co pozwala na bezproblemowe synchronizowanie zmian i ich natychmiastowe testowanie w kontekście rzeczywistej gry. W projekcie wykorzystujemy również funkcje Visual Studio związane z zarządzaniem wersjami kodu, co jest istotne w pracy zespołowej. Kontrola wersji pozwala na równoczesne pracowanie nad różnymi aspektami projektu oraz na bezpieczne wprowadzanie zmian, co minimalizuje ryzyko konfliktów i utraty danych.

4. SPECYFIKACJA I ANALIZA WYMAGAŃ

Niniejszy rozdział zawiera specyfikacje i podstawowe założenia gry komputerowej GPT Chronicles mającej na celu zbadanie potencjalnego wykorzystania dużego modelu językowego do generowania dynamicznych dialogów postaci niezależnych.

4.1. Specyfikacja wymagań (Lange)

4.1.1. Cel i przeznaczenie gry

Celem projektu GPT Chronicles jest stworzenie gry komputerowej, w której dialogi postaci niezależnych są w całości generowane za pomocą dużego modelu językowego. Ponadto, każda postać niezależna musi działać zgodnie z zasadami świata przedstawionego, posiadać pamięć o wcześniejszych konwersacjach z graczem, jego osiągnięciach, a także mieć możliwość wykonywania określonych mechanik gry. Stworzona gra zostanie wykorzystana do zbadania możliwości wykorzystania dużych modeli językowych do dynamicznego generowania dialogów.

4.1.2. Kontekst realizacji

Do stworzenia gry zostaną wykorzystane silnik Unity (por. 3.2) oraz model językowy GPT firmy OpenAI, dostępny poprzez jej API. Wykorzystana będzie także technologia Unity Cloud oferowana przez firmę Unity Technologies oraz darmowe i płatne zasoby z Unity Asset Store. Projekt zostanie zrealizowany i przetestowany przy użyciu komputerów PC z systemem Windows 11. Językiem programowania wykorzystywanym w projekcie będzie C#, natomiast środowiskiem programistycznym – Visual Studio 2022.

4.1.3. Charakterystyka gry

GPT Chronicles będzie grą RPG rozgrywającą się w czasie rzeczywistym, w której gracz wciela się w bohatera z perspektywy trzeciej osoby. Cała akcja toczy się w trójwymiarowym świecie, a poruszanie się w przestrzeni odbywa się za pomocą myszki. Bohater może zmierzyć się z potworami oraz wykonywać zadania dla postaci niezależnych. Grupą docelową są osoby w wieku powyżej 16 lat oraz wszyscy gracze zainteresowani gatunkami RPG i MMORPG, a także innowacyjnymi mechanikami interakcji z postaciami niezależnymi. Unikalnymi cechami gry będą dialogi generowane przez duży model językowy, co umożliwia niepowtarzalne doświadczenie dla każdego gracza. Sztuczna inteligencja podejmuje również decyzje dotyczące wykonywania określonych mechanik gry.

4.1.4. Środowisko pracy i sposób rozpowszechniania

Gra komputerowa przeznaczona na platformę PC z systemem Windows o architekturze Intel 32-bit. Wymagany będzie stały dostęp do Internetu podczas rozgrywki ze względu na konieczność stałego połączenia z API OpenAI oraz serwerami chmury Unity Cloud. Rozpowszechnianie gry odbędzie się poprzez Internet przez bezpośrednie przekazywanie skompresowanej paczki z plikami gry oraz przy pomocy platformy itch.io.

4.1.5. Ograniczenia

- Gra będzie dostępna wyłącznie w języku angielskim.
- Maksymalny czas rozgrywki: 30 minut.
- Czas potrzebny na wypełnienie ankiety: 10 minut.
- Gra wymaga stałego dostępu do Internetu podczas rozgrywki

4.2. Wymagania funkcjonalne (Lange)

W tym podrozdziale przedstawiona jest szczegółowa analiza wymagań funkcjonalnych projektu gry. Jest to zbiór funkcjonalności, które system musi implementować, aby móc spełnić oczekiwany interakcje użytkownika, a tym samym założone cele badawcze.

Wymagania zostały podzielone na siedem kategorii obejmujących różne aspekty gry: dialogi, postacie niezależne, treść i świat gry, dźwięk, zadania, statystyki i system walki oraz inne funkcje. Każda z kategorii zawiera szczegółowe wymagania, które razem tworzą całkowity obraz funkcjonalności systemu. Największy nacisk został położony na aspekty projektu bezpośrednio związane z tematem badawczym, są to więc system dialogów, dynamiczne podejmowanie decyzji przez NPC oraz interaktywność komunikacji.

4.2.1. Dialogi

1. Generowanie dialogów
 - Dialogi w całości generowane za pomocą modelu GPT.
2. Komunikacja z NPC
 - Możliwość komunikacji z NPC za pomocą:
 - Tekstu pisanego
 - Mikrofonu
 - Opcji dialogowych dynamicznie generowanych przez model GPT
3. Możliwości NPC (decyzje podejmowane przez model GPT)
 - Dołączenie do drużyny gracza
 - Przekazanie zadania do wykonania
 - Potwierdzenie wykonania zadania
 - Zaprowadzenie gracza w konkretne miejsce

4.2.2. Postacie niezależne

4. Liczba postaci niezależnych
 - Co najmniej 6 postaci niezależnych.
5. Charakterystyka NPC
 - Każda postać niezależna musi posiadać:
 - Swoją historię
 - Cele
 - Emocje
 - Cechy charakteru
6. Pamięć NPC

- Każda postać niezależna musi pamiętać poprzednie konwersacje z graczem oraz znać jego aktualne osiągnięcia i cele do wykonania.

4.2.3. Treść i świat gry

7. Filtrowanie treści

- System filtrowania treści oparty o model GPT sprawdzający zgodność dialogu pomiędzy graczem i NPC ze światem przedstawionym.

8. Otwarty świat gry

- Gracz może dowolnie poruszać się po otwartym świecie gry.

4.2.4. Dźwięk

9. Udźwiękowienie

- Pełne udźwiękowienie gry, obejmujące efekty dźwiękowe oraz muzykę.

4.2.5. Zadania

10. System zadań

- System wykonywania zadań nadający cel rozgrywki i zintegrowany z generowanymi dialogami.

11. Liczba zadań

- Co najmniej 6 zadań do wykonania.

12. Sprawdzanie zadań

- Możliwość sprawdzenia aktualnego zadania i celu do wykonania.

4.2.6. Statystyki i system walki

13. System statystyk

- System statystyk gracza i przeciwników.

14. Sprawdzanie stanu zdrowia

- Możliwość sprawdzenia obecnego stanu zdrowia gracza i przeciwnika.

15. System walki

- System walki umożliwiający pokonywanie wrogów, oparty o statystyki bohatera i przeciwników.

16. Rodzaje przeciwników

- Co najmniej 3 rodzaje przeciwników.

17. Odrodzenie przeciwników

- Przeciwnicy automatycznie odradzają się po ich śmierci.

18. Odrodzenie gracza

- Odrodzenie postaci gracza po jego śmierci.

4.2.7. Inne funkcje

19. Zatrzymanie rozgrywki

- Możliwość zatrzymania rozgrywki na każdym etapie gry.

20. Informacje o grze

- Możliwość sprawdzenia:

- Sterowania

- Celu projektu badawczego
 - Ogólnego opisu gry
21. Ankietyzacja
- Możliwość wypełnienia ankiety w grze zaraz przed jej wyłączeniem.

4.3. Wymagania niefunkcjonalne (Gafke)

W tym podrozdziale przedstawiona jest szczegółowa analiza wymagań niefunkcjonalnych projektu gry. Wymagania te nie odnoszą się bezpośrednio do poszczególnych funkcji systemu, mają jednak istotne znaczenie dla jego ogólnej jakości, wydajności i użyteczności.

4.3.1. Wydajność

1. Płynność działania
 - Gra musi działać płynnie, zapewniając średnią liczbę klatek na sekundę powyżej 30 na komputerze spełniającym minimalne wymagania sprzętowe:
 - Procesor: Intel(R) Core(TM) i3-2100 CPU @ 3.10GHz (4 CPUs), 3.1GHz
 - Pamięć: 2GB RAM
 - Grafika: Intel(R) HD Graphics 520
 - Rozmiar na dysku: 250MB
2. Szybkość generowania dialogów
 - Czas potrzebny do wyświetlenia wygenerowanego dialogu nie powinien przekraczać 2 sekund.

4.3.2. Kompatybilność

3. Wsparcie dla platformy PC
 - Gra musi być kompatybilna z systemem operacyjnym Windows 7+ działającym na architekturze Intel 32-bit.
4. Wsparcie dla urządzeń wejściowych
 - Gra musi wspierać korzystanie z klawiatury, myszy oraz mikrofonu.

4.3.3. Użyteczność

5. Czytelność rozgrywki
 - Gracz musi łatwo i szybko zorientować się, jaki cel ma do wykonania oraz jakie interakcje może przeprowadzić w grze.
6. Prosty i czytelny interfejs użytkownika
 - Interfejs użytkownika musi być intuicyjny, prosty w obsłudze i czytelny dla gracza.

4.3.4. Niezawodność

7. Brak krytycznych błędów
 - Gra musi być pozbawiona błędów, które uniemożliwiają rozgrywkę.

4.3.5. Przechowywanie danych

8. Zapisywanie danych z gry

- Dane gry, w tym dane dialogowe, muszą być zapisywane w usłudze chmurowej Unity Cloud w formacie JSON.
9. Przechowywanie danych z ankiet
- Dane z ankiet muszą być przechowywane w chmurze w formacie CSV.

4.3.6. Integracja z zewnętrznymi systemami

10. Komunikacja z API OpenAI
- Gra musi komunikować się z API OpenAI przy pomocy REST API, używając formatu JSON.
11. Komunikacja z API Unity Cloud
- Gra musi komunikować się z API Unity Cloud przy pomocy REST API, używając formatu JSON.

4.3.7. Bezpieczeństwo i prywatność

12. Anonimowość użytkownika
- Dane z gry nie mogą być powiązane z żadną osobą ani lokalizacją, zapewniając pełną anonimowość użytkownika.
13. Anonimowość ankietyzacji
- Dane uzyskane z badań i ankiet będą wykorzystane wyłącznie w celach naukowych i usunięte po zakończeniu badań.

4.4. Kryteria akceptacji (Gaffke)

Kryteria akceptacji będą kluczowe dla oceny gotowości gry do jej rozpowszechnienia w celu przeprowadzenia badań. Proces akceptacji obejmuje zarówno testy deweloperskie, jak i oceny przeprowadzane przez promotorą.

4.4.1. Testy alfa

Testy alfa przeprowadzone przez deweloperów w celu wstępnej oceny jakości, wydajności i zgodności z wymaganiami. Kryteria akceptacji obejmują:

1. Kompletność funkcji
 - Wszystkie wymagane funkcjonalności gry, w tym generowanie dialogów, interakcje z NPC, systemy zadań i walki, muszą być całkowicie zaimplementowane i dostępne do testów.
2. Plynność działania
 - Gra musi działać płynnie, zapewniając średnią liczbę klatek na sekundę powyżej 30 na komputerach o minimalnych wymaganiach sprzętowych.
3. Brak krytycznych błędów
 - Gra nie może zawierać błędów krytycznych uniemożliwiających rozgrywkę lub powodujących awarie.
4. Spójność dialogów
 - Dialogi generowane przez GPT muszą być spójne ze światem gry, sensowne i dostosowane do kontekstu interakcji.
5. Interaktywność i pamięć NPC
 - NPC muszą prawidłowo reagować na interakcje gracza, pamiętać poprzednie konwersacje oraz znać aktualne osiągnięcia i cele gracza.

6. Kryterium ukończenia gry
 - Możliwe jest ukończenie gry od początku do końca wraz z wykonaniem wszystkich zadań.
7. System filtracji treści
 - System filtrowania treści musi skutecznie sprawdzać zgodność dialogów ze światem gry i eliminować nieodpowiednie dane wejściowe.
8. Funkcjonalność ankiet
 - Ankiety w grze muszą być w pełni funkcjonalne, a dane z ankiet muszą być prawidłowo zapisywane w chmurze Unity Cloud w formacie CSV.
9. Dostępność i użyteczność
 - Gra musi być intuicyjna i łatwa w obsłudze, mieć czytelny interfejs użytkownika i informacje dla nowego gracza.

4.4.2. Walidacja promotora oraz graczy

Po ukończeniu testów alfa gra zostanie przekazana promotorowi oraz czterem zaprzyjaźnionym graczom, w celu oceny jej pod kątem spełnienia celów badawczych i zgodności z wymaganiami projektu magisterskiego.

1. Zgodność z celami badawczymi
 - Gra musi skutecznie demonstrować wykorzystanie dużych modeli językowych do generowania dialogów NPC.
2. Jakość i spójność dialogów
 - Dialogi w grze muszą być ocenione jako wysokiej jakości, sensowne i spójne ze światem gry.
3. Pełność funkcjonalności
 - Wszystkie zaplanowane funkcje i mechaniki gry muszą być zaimplementowane i działające zgodnie z opisanymi wymaganiami funkcjonalnymi i niefunkcjonalnymi.
4. Dane z ankiet
 - Ankiety muszą dostarczać wartościowych danych badawczych umożliwiających ocenę efektywności wykorzystania modeli językowych w grze.
5. Dokumentacja techniczna
 - Gra musi być dobrze udokumentowana, z pełnym opisem technicznym implementacji, użytych technologii i przeprowadzonych testów.

4.4.3. Testy beta

Po wstępnej walidacji gra zostanie udostępniona ograniczonej grupie testerów, którzy ocenią ją pod kątem funkcjonalności w realnych warunkach.

1. Pozytywne opinie testerów
 - Gra musi uzyskać pozytywne opinie testerów, w szczególności pod kątem płynności działania, grywalności i jakości dialogów.
2. Zbieranie danych z ankiet
 - Dane z ankiet przeprowadzonych przez testerów muszą być użyteczne w kontekście projektu badawczego.
3. Identyfikacja i naprawa błędów
 - Wszelkie zgłoszone problemy lub błędy muszą być naprawione.

4.5. Zespół projektowy (Lange)

Zespół projektowy implementujący system składa się z dwóch deweloperów będących równocześnie autorami owej pracy. Obaj mają doświadczenie zawodowe w programowaniu oraz podłożę teoretyczne w dziedzinie projektowania gier. Poniżej znajduje się podział ról w projekcie:

Deweloper Florian Gaffke

Kierownik projektu / Programista / Projektant interfejsu użytkownika / Tester

Odpowiedzialny za zarządzanie projektem, koordynacje działań oraz komunikacje z promotorem. Ponadto realizator kluczowej mechaniki gry, systemu dialogowego, a także systemu statystyk oraz systemu walki. Zajął się również tworzeniem interfejsu użytkownika oraz testowaniem funkcji.



Deweloper Karol Lange

Programista / Projektant graficzny / Projektant dźwięku / Tester

Odpowiedzialny za projekt wizualny gry, tworzenie postaci, środowisk, a także ścieżki dźwiękowej oraz efektów dźwiękowych. Realizator mechanik interakcji, poruszania się, systemu zadań oraz sztucznej inteligencji przeciwników i NPC. Zajął się również zarządzaniem systemem kontroli wersji, serwerem Unity Cloud oraz testowaniem funkcji.



5. PROJEKT GRY GPT CHRONICLES

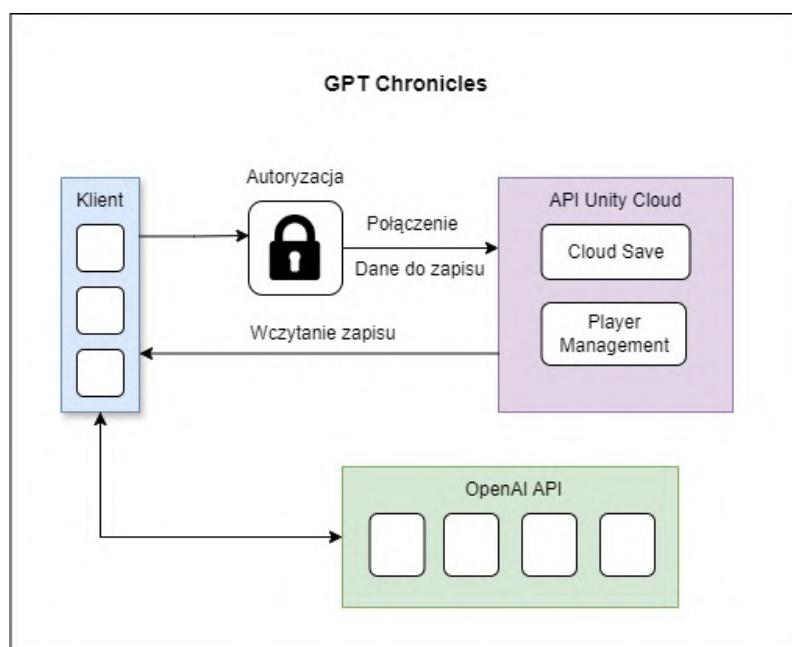
W tym rozdziale przedstawiony zostanie projekt gry GPT Chronicles wraz z opisem technicznym jej implementacji.

5.1. Architektura gry (Lange)

W projekcie systemu wykorzystano kilka wzorców architektonicznych oraz zasad projektowych, aby efektywnie wspierać złożone interakcje charakterystyczne dla gier RPG, a także dynamiczne dialogi, które opierają się na modelu GPT, stanowiącym centralny element rozgrywki. Poniżej przedstawiono kluczowe elementy oraz strukturę zaimplementowaną w systemie, wraz z przykładami ich implementacji.

5.1.1. Diagram architektury

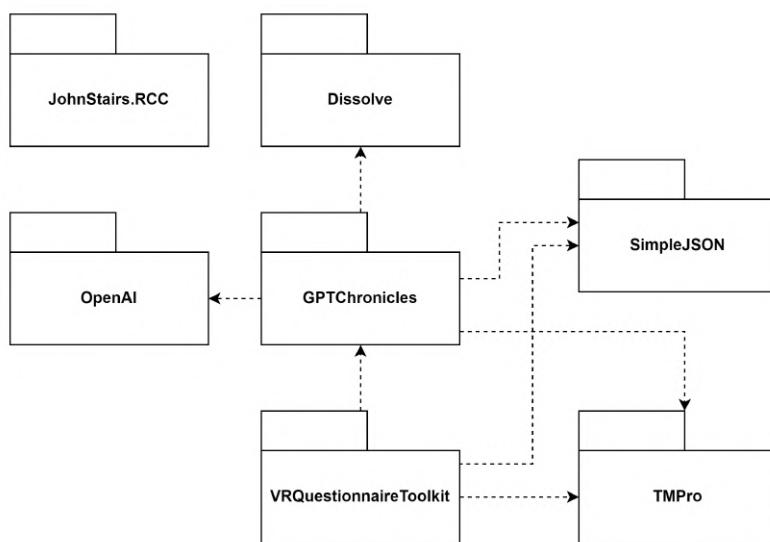
Na rys. 5.1 przedstawiony został wysokopoziomowy diagram architektury. Cały system składa się z trzech komunikujących się podsystemów. Najważniejszym elementem jest Klient gry, jest to aplikacja komputerowa przeznaczona na systemy Windows, która powstała po zbudowaniu projektu Unity. Po uruchomieniu gry, następuje komunikacja poprzez sieć z serwerami Unity Cloud. Aby tego dokonać niezbędna jest autoryzacja użytkownika, która dokonywana jest automatycznie przy starcie gry, również wtedy pobierane są wszelkie zapisane dane, które mogą znajdować się w chmurze. Podczas gry Unity Cloud wielokrotnie obsługuje również zapis danych, są to m.in. informacje o stanie wykonanych zadań, zapisy dialogów czy ankiety wypełnione przez graczy. Trzecim podsystemem są serwery firmy OpenAI udostępniające API modelu językowego GPT, w trakcie gry po wykonaniu odpowiedniej interakcji przez gracza następuje wywołanie zapytania do API i obsługa wiadomości zwrotnych.



Rysunek 5.1: Diagram architektury

5.1.2. Diagram pakietów

Rys. 5.2 przedstawia diagram pakietów wizualizujący podstawowe zależności pomiędzy systemem GPT Chronicles, a komponentami zewnętrznymi wykorzystywanymi w projekcie. Najważniejszym komponentem zewnętrznym jest paczka Unity (ang. *Unity Package*) OpenAI, która udostępnia interfejs za pomocą, którego można łatwo komunikować się z API OpenAI. Jest on szeroko używany w systemie dialogowym. Kolejną paczką jest VRQuestionnaireToolkit, który udostępnia skrypty i elementy interfejsu użytkownika pozwalające na tworzenie zróżnicowanych ankiet we wnętrzu gry w sposób zautomatyzowany. Pakiet JohnStair.RCC odpowiada za obsługę kamery gracza, a Dissolve zawiera w sobie szereg prefabów odpowiedzialnych za efekty specjalne. TMPPro jest systemem, który można dołączyć do projektu Unity udostępniającym szerokie możliwości obsługi tekstu wykorzystywanego w elementach interfejsu użytkownika, w tym rozbudowane formattowanie, edytowanie wyglądu i inne. Ostatni element to biblioteka SimpleJSON służąca do obsługi plików JSON i pozwalająca na łatwą manipulację i przetwarzanie danych. Wykorzystana ze względu na dane zapisywane w chmurze Unity Cloud, które są przechowywane w tym formacie.



Rysunek 5.2: Diagram pakietów

5.1.3. Diagram komponentów składowych

Diagram znajdujący się na rys. 5.3 przedstawia komponenty składowe systemu. Ze względu na rozbudowanie systemu został on zgeneralizowany, aby w prostszy i klarowniejszy sposób wyjaśnić ogólną architekturę i zależności pomiędzy systemami tworzącymi grę.

Fundamentem gry jest silnik Unity, który zapewnia podstawowe funkcjonalności niezbędne do jej działania. Przede wszystkim dostarcza on menedżer wejścia (ang. *Input*) odpowiedzialny za rejestracje i przetwarzanie danych wejściowych od gracza, takich jak naciśnięcia klawiszy i ruchy myszy. Dalej jest renderer grafiki i silnik fizyki umożliwiające prezentację wizualną świata gry oraz symulacje fizyki umożliwiającą ruch obiektów i interakcje między nimi. System audio pozwala na zarządzanie dźwiękami i muzyką, a system NavMesh umożliwia nawigację AI w świecie gry, szczególnie istotną dla ruchu gracza oraz NPC.

Główne komponenty gry odpowiadają za zarządzanie logiką, są to różnego rodzaju menedżerowie specjalizujący się w rozmaitych aspektach systemu. Najważniejszy jest menedżer gry (ang. *Game Manager*) nadzorujący ogólny stan gry, koordynuje on inne systemy.

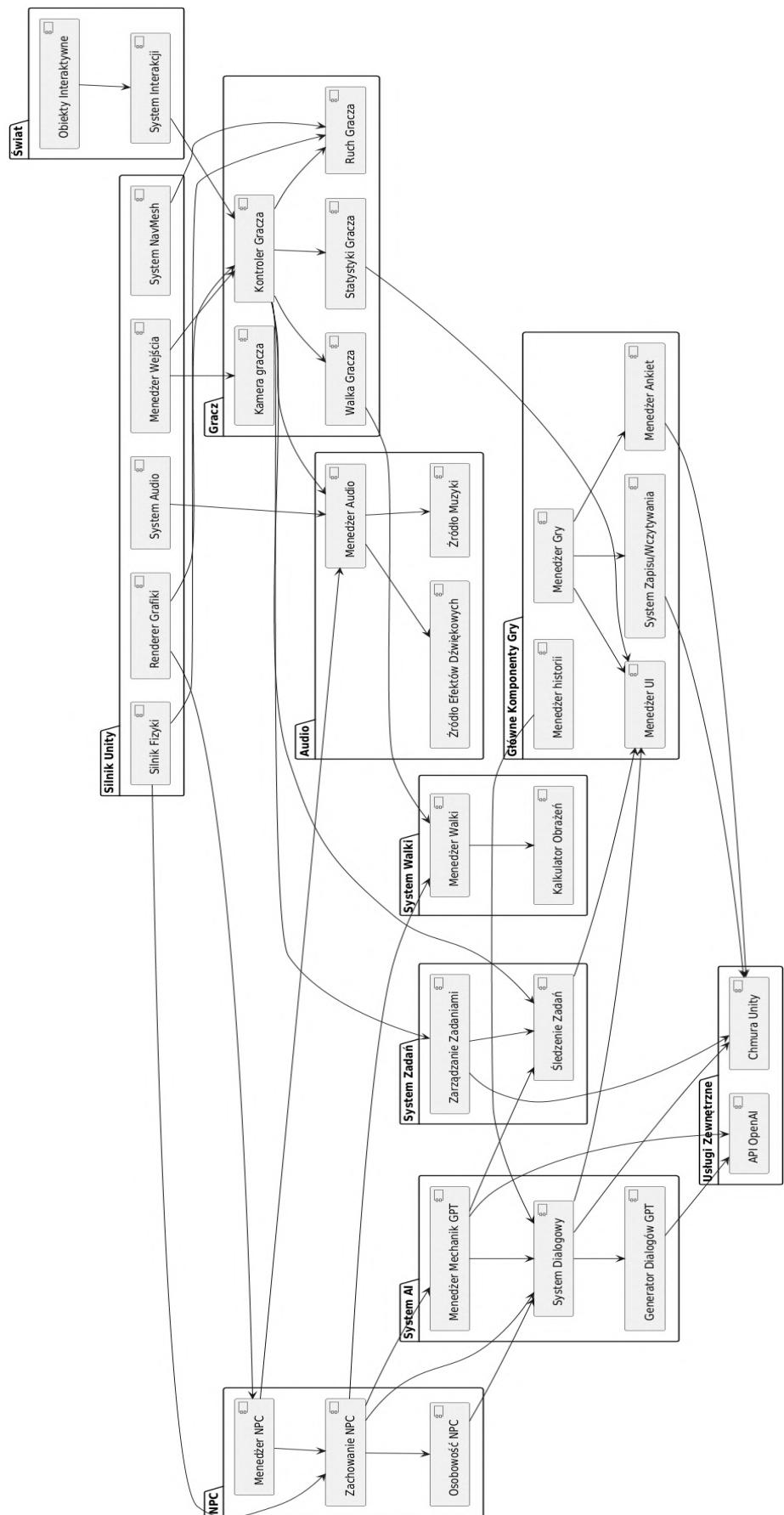
- Menedżer ankiet odpowiada za obsługę ankiet zbieranych od graczy.
- Menedżer UI zarządza interfejsem użytkownika, wyświetla odpowiednie informacje i umożliwia interakcje z jego elementami.
- Menedżer historii zawiera wszystkie najważniejsze informacje o świecie przedstawionym i udostępnia metody umożliwiające aktualizowanie tej wiedzy, służy on do przekazywania tych informacji pomiędzy poszczególnymi NPC w systemie dialogowym.
- System zapisu i wczytywania odpowiada za komunikację z serwerem Unity Cloud i odpowiednie zapisywanie i wczytywanie stanu gry oraz danych badawczych.

"Gracz" zawiera komponenty związane bezpośrednio z postacią sterowaną przez gracza. Głównym komponentem jest kontroler gracza (*ang. Player Controller*) sterujący wszystkimi aspektami postaci gracza, w tym ruchem, walką, interakcjami z otoczeniem i statystykami. Gracz może wchodzić w interakcje z obiektami w świecie gry. Obiekty interaktywne (*Interactable*) definiują interaktywne elementy w grze, a system interakcji zarządza zależnościami między graczem i tymi obiektami. Pozwala to na takie działania jak rozmowa z NPC, walka czy wywoływanie zdarzeń. Postacie niezależne i przeciwnicy mają swój oddzielny system NPC. Zarówno przeciwnicy jak i NPC są obsługiwani tym samym systemem, to system interakcji decyduje czy gracz w trakcie interakcji podejmie walkę czy dialog. Zachowanie NPC określa jak zachowują się oni w świecie gry, wpływają na nie działania gracza, decyzje modelu GPT, system fizyki i NavMesh (ruch po mapie). Z kolei osobowość NPC nadaje każdej postaci niezależnej unikalne cechy, charakter i historię wykorzystywaną przez system dialogowy. Interakcja między graczem, a NPC (lub NPC i NPC) może prowadzić do dwóch rezultatów, pierwszym jest walka obsługiwana przez system walki. Walka polega jedynie na wymianie ciosów pomiędzy postaciami biorącymi w niej udział, a obrażenia są obliczane przez komponent kalkulator obrażeń biorący pod uwagę statystyki obu uczestników. Drugą możliwością jest dialog będący elementem systemów AI, gdyż opiera się na automatycznym generowaniu tekstu przez duży model językowy. System dialogowy wykorzystuje komponent komunikujący się z API OpenAI, menedżerem historii i osobowością NPC, z którym prowadzona jest interakcja do generowania adekwatnych do sytuacji i realistycznych linii dialogowych. Jest on również połączony z systemem mechanik NPC, które podejmowane są na podstawie prowadzonej konwersacji, są to system zadań, możliwość dołączenia do drużyny gracza, czy też możliwość zaprowadzenia gracza w jakieś miejsce. Za oprawę dźwiękową odpowiada menedżer audio komunikujący się z dwoma głównymi źródłami dźwięku odpowiadającymi za efekty dźwiękowe i muzykę, a także ze źródłami dźwięku dołączonymi do obiektów gracza i NPC.

Ogólny przepływ mechanik wygląda zatem następująco:

1. Menedżer wejścia przekazuje dane do kontrolera gracza, który zarządza ruchem, walką i statystykami gracza.
2. System interakcji umożliwia graczowi interakcję z obiektami interaktywnymi w świecie gry, w tym NPC i przeciwnikami.
3. System AI, w tym generator dialogów i menedżer mechanik GPT poprzez współpracę z systemem dialogowym i komponentami NPC tworzy dynamiczne dialogi i interakcje.
4. Zarządzanie zadaniami odbywa się poprzez współpracę z kontrolerem gracza, systemem mechanik NPC i menedżerem UI, pozwala ono na dostarczanie i monitorowanie misji.

5. System walki koordynuje walkę pomiędzy graczem a NPC (lub NPC i NPC) wykorzystując do tego statystyki i kalkulator obrażeń.
6. System jest zintegrowany z usługami zewnętrznymi, API OpenAI jest wykorzystywane przez system dialogowy i mechaniki NPC, podczas gdy chmura Unity Cloud służy do zapisu i odczytu wszystkich potrzebnych danych.
7. Dźwięk jest zarządzany przez menedżer audio, dwa główne komponenty audio oraz inne komponenty audio dołączone do różnych obiektów w świecie gry.



Rysunek 5.3: Diagram komponentów

5.1.4. Struktura projektu

Na rys. 5.4 przedstawiono strukturę projektu gry, która obejmuje wszystkie utworzone skrypty. Aby zapewnić jak najbardziej wiarygodne wyniki badań, gra zawiera wszystkie kluczowe elementy charakterystyczne dla pełnoprawnej produkcji, takie jak system zadań, podstawowy system walki, system audio i efektów dźwiękowych, pełny interfejs użytkownika, system statystyk gracza, system dialogowy oparty na sztucznej inteligencji, który jest głównym przedmiotem badań, a także różne mechaniki NPC, których działanie również zależy od AI. W kolejnych podrozdziałach omówione zostaną poszczególne systemy oraz sposób ich działania. Kluczowymi elementami będącymi obiektem badań w niniejszej pracy są skrypty znajdujące się w katalogu GPT, przy czym za dialogi NPC odpowiadają klasy DialogGPT i HistoryManager. Natomiast za mechaniki postaci niezależnych odpowiedzialne są skrypty StateMachineGPT, GameMasterGPT oraz NPCMechanic.



Rysunek 5.4: Struktura projektu

5.2. Mechaniki gry (*Lange i Gaffke*)

Gra komputerowa składa się z wielu podsystemów, które współpracują ze sobą w celu stworzenia spójnej rozgrywki. W niniejszym rozdziale przedstawione zostaną kluczowe systemy odpowiedzialne za podstawowe mechaniki gry, są to:

- System poruszania się
- System interakcji
- System dialogowy oparty na sztucznej inteligencji
- Systemy związane z NPC
 - System nawigacji NPC
 - Mechaniki NPC
- System zadań
- System zdrowia i statystyk
- System walki
- System zarządzania audio
- System zarządzania interfejsem użytkownika

W tym rozdziale opisane zostaną budowa wszystkich tych systemów oraz przykłady implementacji wybranych funkcji.

System poruszania się (*Lange*)

Gra GPT Chronicles jest grą RPG, w której gracz kontroluje swoją postać z widoku trzeciej osoby. System nawigacji jest inspirowany grą Lineage II, popularną grą komputerową z gatunku MMORPG stworzoną przez koreańską firmę NCsoft w 2003 roku. Jest to system bardzo intuicyjny i oparty na klasycznej mechanice "*click to move*". Całość nawigacji odbywa się za pomocą myszy. Gracz przemiesza swoją postać poprzez kliknięcie lewym przyciskiem myszy w dowolnym punkcie na mapie, co natychmiast powoduje, że postać zaczyna zmierzać w to miejsce. Implementacji tego systemu dokonano przy pomocy wbudowanego w Unity systemu NavMesh. NavMesh to siatka nawigacyjna, która automatycznie generuje się na podstawie topologii terenu i innych obiektów na scenie. Definiuje ona obszary, po których mogą poruszać się agenci, a także miejsca niedostępne takie jak zbocza gór, ściany czy drzewa. Przykładem takiego agenta jest postać gracza, przeciwnicy i postacie niezależne. Dzięki temu gracz może kliknąć w dowolne miejsce, a sztuczna inteligencja zaimplementowana w silniku Unity automatycznie wytypuje ścieżkę, która pozwoli na dotarcie do tego celu. Za całą obsługę tego elementu rozgrywki odpowiada komponent PlayerController.

Kamera w GPT Chronicles również działa w taki sam sposób jak w grze Lineage II. Gracz ma możliwość wykonania dwóch interakcji z kamerą, pierwsza wykonywana jest za pomocą kółka myszy i pozwala na zbliżenie i oddalenie kamery umożliwiając tym samym spojrzenie na szczegóły otoczenia lub zyskanie szerszej perspektywy podczas eksploracji świata. Druga interakcja wykonywana jest prawym przyciskiem myszy, po jego przytrzymaniu gracz może swobodnie obracać kamerę wokół postaci, co daje pełną kontrolę nad kątem widzenia. Do implementacji kamery wykorzystano istniejący zasób z Unity Asset Store o nazwie *RPG Cameras & Controllers*. Zawiera

on paczkę skryptów i pozwala na bardzo łatwą implementację opisanego zachowania kamery poprzez dołączenie do postaci dwóch komponentów. Pierwszym jest RPGViewFrustum pozwalający na dokładne dostosowanie efektów wizualnych takich jak kształt pola widzenia, okluzję kamery, zanikanie obiektów zasłaniających widok i wiele innych. Drugim komponentem jest RPGCamera, który odpowiada za techniczne ustawienia takie jak zasięg przybliżenia i oddalenia, szybkość poruszania, ustawienia kurSORA i przycisków myszy czy interakcje z wodą i innymi substancjami.

System interakcji (*Lange*)

System interakcji to kluczowy system, który pozwala graczowi na oddziaływanie na różnorodne elementy środowiska takie jak NPC czy przeciwnicy. Poniżej znajduje się dokładny opis tego jak system działa wraz z kluczowymi klasami i metodami.

Kluczowe komponenty:

1. PlayerController - zarządza ruchem gracza, interakcją i aktualizacją stanów.
2. Interactable - reprezentuje obiekty, z którymi gracz może przeprowadzić interakcje takie jak przeciwnicy i NPC.
3. DialogWindow - element interfejsu użytkownika do wyświetlania dialogów
4. InteractableType - Enum zawierający typy obiektów interaktywnych
5. NavMeshAgent - Umożliwia znajdowanie ścieżki, aby gracz mógł się poruszać
6. TargetWindowParent i TargetWindow - zarządza interfejsem użytkownika dla wybranego celu
7. GameEventManager - zarządza wydarzeniami wewnętrz gry

Na rys. 5.5 widoczny jest przepływ procesu opisujący działanie mechaniki interakcji. W trakcie gry gracz poprzez zmianę ikony kurSORA myszy przy najechaniu na odpowiednie obiekty w środowisku 3D może podjąć z nimi interakcję. Po kliknięciu na cel wyświetla się element interfejsu użytkownika w górnej części ekranu wskazujący na zaznaczenie celu przez gracza. Po ponownym kliknięciu na cel następuje automatyczne wyznaczenie ścieżki do celu poprzez metody klasy NavMeshAgent wykonane w klasie PlayerController i metodzie ClickToMove widoczne we fragmencie kodu 5.1. W momencie zbliżenia się na odpowiednią odległość wykonywana jest metoda ReachedDestination, w której podejmowana jest decyzja o typie interakcji, jeśli obiekt posiada InteractableType NPC wtedy otwierane jest okno dialogowe, natomiast jeśli jest przeciwnikiem rozpoczyna się walka. W grze istnieje trzeci typ interakcji nazwany Item jednak nie jest on używany w wersji końcowej gry.



Rysunek 5.5: Diagram przepływu mechaniki interakcji

```

private void ClickToMove() {
    if (!EventSystem.current.IsPointerOverGameObject() &&
        !playerActor.GetIsDead()) {
        RaycastHit hit;
        if (Physics.Raycast(Camera.main.ScreenPointToRay(Input.mousePosition),
            out hit, 100, clickableLayers)) {
            if (hit.transform.CompareTag("Interactable")) {
                currentTarget = hit.transform.gameObject;
                ShowTargetWindow();
                Interactable interactable =
                    currentTarget.GetComponent<Interactable>();
                // Visual effects and targeting logic
            } else {
                if (agent.enabled)
                {
                    NavMeshPath path = new NavMeshPath();
                    agent.CalculatePath(hit.point, path);
                    agent.path = path;

                    //Click effect
                }
            }
        }
    }
}

```

Listing 5.1: Metoda ClickToMove

System dialogów (Gaffke)

Najważniejszym systemem w grze i jednocześnie systemem będącym obiektem badań niniejszej pracy jest system dialogów, który funkcjonuje w pełnym oparciu o duży model językowy GPT-4o i GPT-4o mini. System zawiera wiele komponentów, w tym elementy zarządzające interfejsem użytkownika, generowaniem odpowiedzi przez AI czy obsługą trzech form komunikacji, z których może skorzystać gracz.

W przypadku systemu dialogów kluczowymi komponentami są:

1. DialogWindow
2. DialogGPT - obsługuje generowanie dialogów z wykorzystaniem modeli GPT.
3. StateMachineGPT - obsługuje podejmowanie decyzji o wykonaniu mechanik NPC również z użyciem modelu językowego GPT.
4. GameMasterGPT - obsługuje różnorodne kwestie także z użyciem sztucznej inteligencji, między innymi generowanie opcji dialogowych, ocenę czy dialog jest zgodny ze światem przedstawionym czy generowanie opisów zadań.
5. NPCAI - zarządza zachowaniami NPC na podstawie przeprowadzonego dialogu.
6. Unity Cloud Save - zarządza zapisem i odczytem historii dialogów

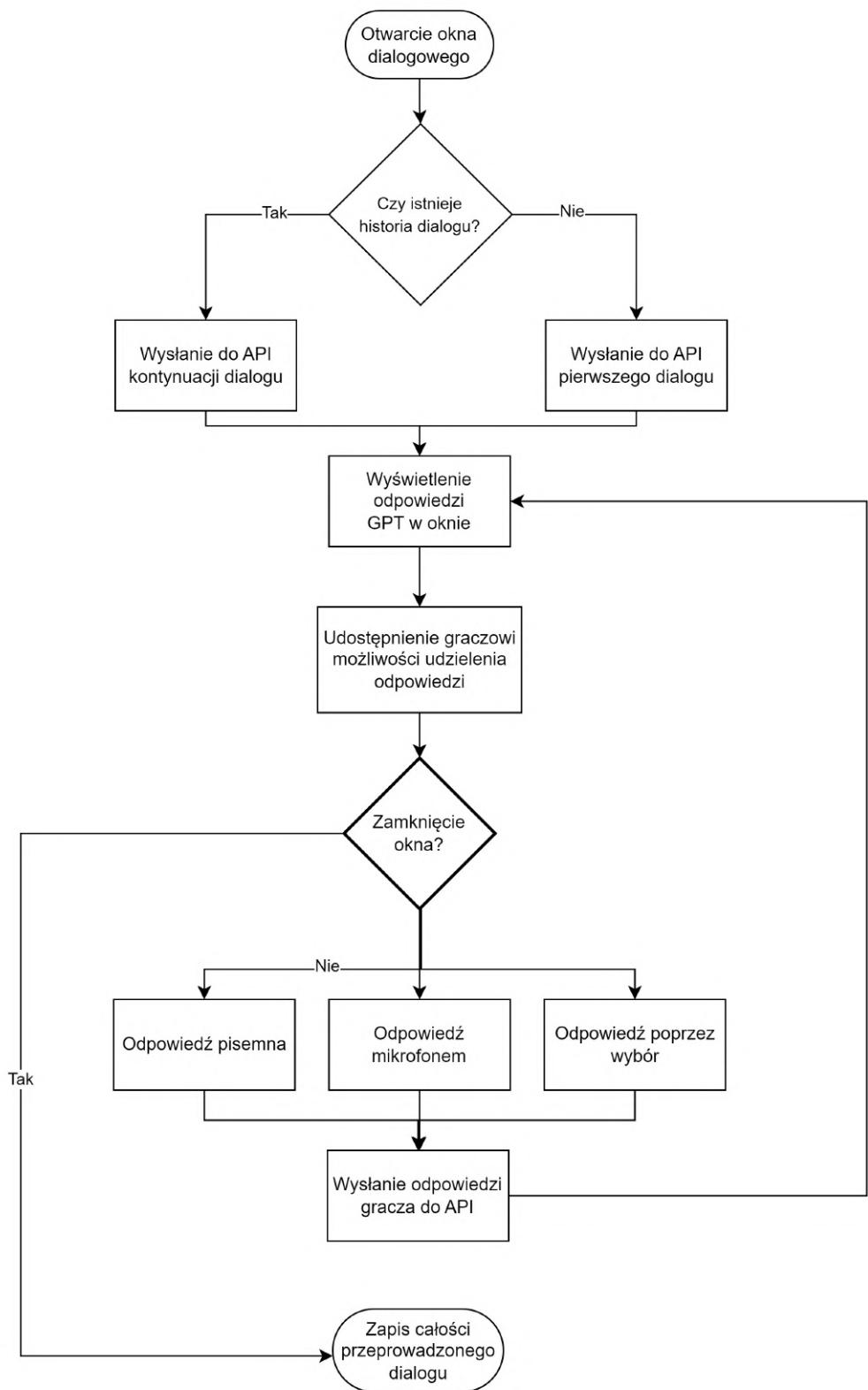
7. HistoryManager - zarządza podstawowymi informacjami o świecie i wydarzeniami

8. Personality - zawiera osobowość i historię NPC

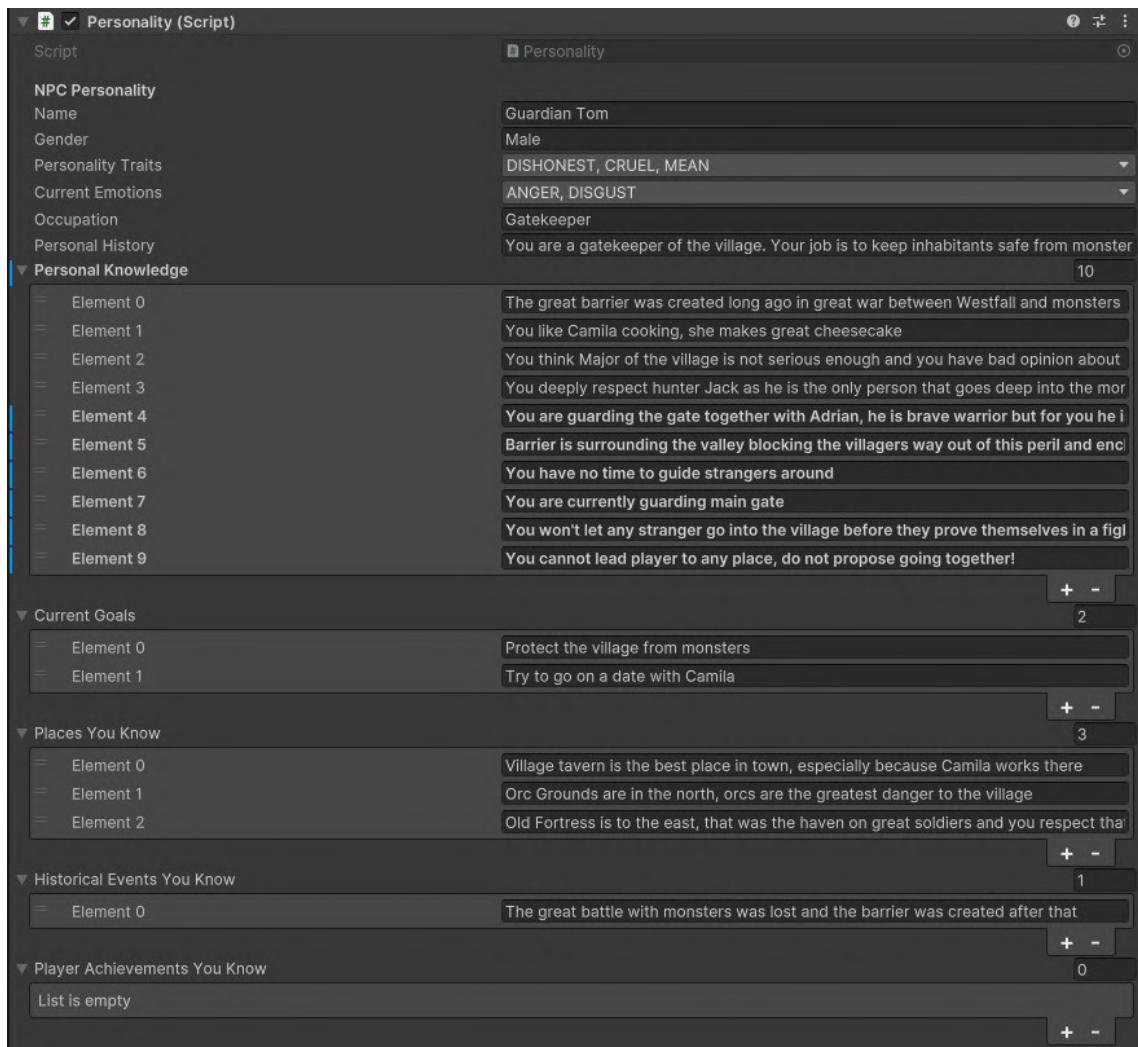
Ogólny diagram przepływu dialogu jest widoczny na rys. 5.6. Mechanika zaczyna się w momencie otworzenia się okna dialogowego, którego można dokonać poprzez mechanikę interakcji z postacią NPC. Pierwszą rzeczą po otwarciu okna jest natychmiastowa i automatyczna próba pobrania z zapisu chmurowego historii dialogów z tym konkretnym NPC. Jeśli takowy istnieje wykonuje się metoda ContinueDialog, w przeciwnym wypadku będzie to metoda InitializeDialog. Obie te metody znajdują się w komponencie DialogGPT. Obie metody działają na podobnej zasadzie, najpierw pobierany jest z NPC komponent Personality widoczny na rys. 5.7 zawierający najważniejsze informacje o danej postaci niezależnej, a także informacje o osiągnięciach gracza i jego obecnych zadaniach. Potem z singletona HistoryManager pobierane są ciągi używane w każdym dialogu bowiem zawierają podstawowe informacje o świecie przedstawionym. Jeśli historia dialogu istniała dodane zostają wszystkie poprzednie dialogi odbyte z tym NPC, w innym wypadku przekazywana jest informacja o pierwszym spotkaniu. Na podstawie tych danych tworzony jest ciąg wejściowy, który zostanie wysłany do API OpenAI (dokładna struktura tego ciągu jest opisana w podrozdziale 5.5). Po uzyskaniu wiadomości zwrotnej od API wraz z wygenerowanym dialogiem zostaje on wyświetlony w oknie dialogowym i gracz uzyskuje możliwość odpowiedzi. Udostępnione zostają trzy propozycje:

1. Gracz może napisać odpowiedź pisemnie przy użyciu pola do wprowadzania tekstu.
2. Możliwość wybrania mikrofonu i nagranie swoich słów, które następnie są konwertowane na tekst do wysłania.
3. Wybór jednej z trzech opcji dialogowych automatycznie wygenerowanych przez model językowy w oparciu o właśnie przeprowadzony dialog i zadania, które gracz ma do wykonania.

Po wysłaniu odpowiedzi wykonywana jest metoda SendReply, która działa na podobnej zasadzie jak dwie poprzednie, jednak tym razem zawiera w sobie odpowiedź gracza. Dialog jest kontynuowany do momentu zamknięcia przez gracza okna dialogowego lub odejścia na odpowiednią odległość od NPC (wtedy okno automatycznie się zamyka). Jeśli tak się stanie cały przeprowadzony dialog jest automatycznie zapisywany do serwisu Unity Cloud Save przy użyciu metody SaveMessageData.



Rysunek 5.6: Diagram przepływu dialogu



Rysunek 5.7: Zawartość komponentu Personality

W momencie prowadzenia dialogu wykonywane są także inne czynności poza generowaniem odpowiedzi również będące elementem systemu i mające za zadanie urozmaicenie rozgrywki. Są to automatycznie generowane opcje dialogowe, mechanizm sprawdzający zgodność odpowiedzi gracza ze światem przedstawionym oraz połączenie dialogów z realnymi akcjami, które mogą wykonywać NPC takie jak wykonywanie zadań, dołączanie do drużyny gracza czy też prowadzenie gracza w różne miejsca. Aby lepiej zrozumieć przebieg wykonywania wszystkich tych operacji należy spojrzeć na diagram pokazujący przebieg metody `SendReply`, która łączy w sobie wszystkie te operacje. Znajduje się on na rys. 5.8. Po wstępny przygotowaniu interfejsu użytkownika poprzez całkowite wyczyszczenie okna i zablokowanie interakcji następuje sprawdzenie zgodności odpowiedzi gracza ze światem przedstawionym. Jest to niezbędna operacja mająca na celu powstrzymanie gracza od próby wprowadzenia NPC z roli na przykład poprzez wpisywanie pytań nie na temat, wpisywanie niemającego sensu tekstu, treści wulgarnych lub innych podobnych treści. Użyta do tego jest metoda `GPTGameMasterDecide`, która na wejściu otrzymuje odpowiedź gracza oraz ostatnią wiadomość z historii dialogu (najczęściej wiadomość NPC). Następnie tworzony jest ciąg wejściowy do modelu GPT 4o mini, który ocenia zgodność tych treści z historią świata, funkcja ta jest widoczna we fragmencie kodu 5.2. Sama instrukcja systemowa tworzona na początku metody wygląda następująco:

You will be provided with the latest dialog messages of the player and the non-player character of this world. You have to follow this step by step instruction:

Step 1 – Interpret the description of the game world and then the dialogue lines

Step 2 – Decide if the latest player sentence is adequate to the game world. Also take into consideration non-player character sentence. Remember that player should not have any knowledge that is outside the presented world. Also notice that player can have a normal small talk with the non-player character if it does not go beyond world boundaries.

Step 3 – Respond with only a single word: PASS if the player sentence is ok to be passed and DECLINE if player sentence is out of world boundaries.

Here is the world description:

This is a video game. The game is like typical RPG game with 3D overworld that player can explore. Player can take certain actions such as talking to non-player characters, initiating battles, and traveling. You can say it's typical fantasy world. The creatures in this game are called Monsters. They can have multiple forms, magical, animal-like or human-like. There are no real animals in this world, the game world is inside a valley that is surrounded by magical barrier which no one can escape. In this world there's no such thing as internet, mobile devices and similar. Player shouldn't know anything about the real world and only exist within the videogame simulation. Player shouldn't break the game world rules. He should not use emojis or smileys. If the player directly responses to non-player character dialog or tries to initiate small talk allow it to pass.

Przykład pełnego ciągu wejściowego znajduje się w podrozdziale 5.5. Dalej przekazywane są ostatnie rozmowy między graczem i NPC. We fragmencie kodu widać również, że model powinien reagować poprzez odpowiedź w postaci pojedynczego tokenu reprezentującego jego ocenę wypowiedzi (podobnie jest w przypadku podejmowania decyzji o wykonywaniu mechanik). Aby dodatkowo zapewnić większą szansę na uzyskanie dobrej odpowiedzi w postaci tokenu i powtarzalności wyników parametr temperatury dla tych operacji został ustawiony na wartość 0.2. W razie, gdyby GPT nie zwrócił pojedynczego wyrazu uznamy, że wypowiedź jest zgodna z zasadami. Jeśli odpowiedź gracza przejdzie zatwierdzenie następnie wysyłana jest do API OpenAI w celu uzyskania odpowiedzi NPC (tak jak opisano wcześniej). Po uzyskaniu odpowiedzi następuje sprawdzenie czy znajduje się w niej intencja wykonania mechaniki NPC. Działa to na podobnej zasadzie jednak tym razem używany jest singleton StateMachineGPT, do którego przekazywana jest ostatnia wypowiedź gracza, ostatnia odpowiedź NPC, mechanika, opis mechaniki i tokeny jakie model może zwrócić w odpowiedzi. Metoda jest niemal identyczna, zmienia się jednak ciąg wejściowy do API:

You will be provided with a description of game mechanic and the last dialog messages of the Player and the non-player character that he is talking to. You have to follow following step by step instruction:

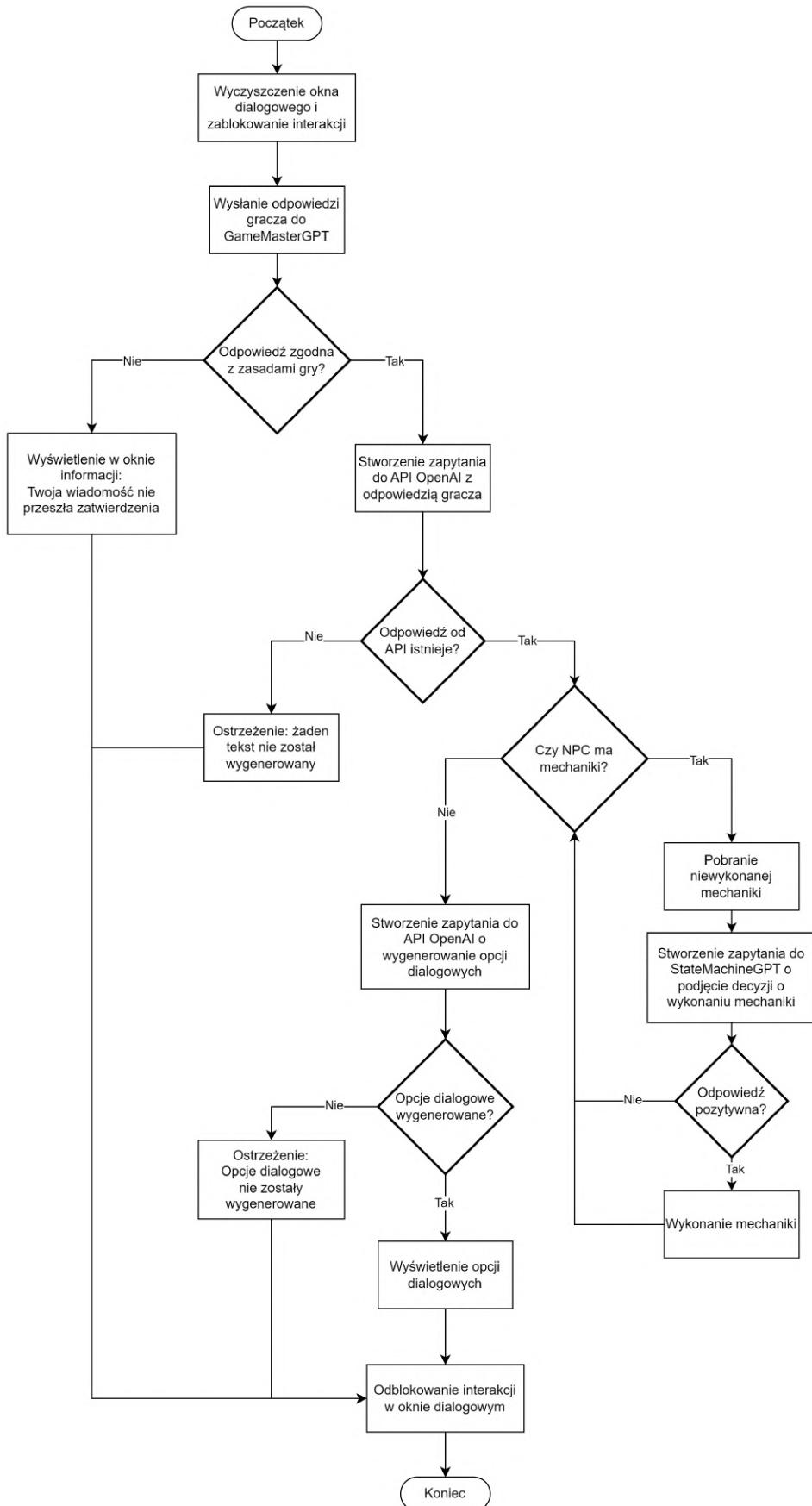
Step 1 – The game mechanic is described by three things. Mechanic, Mechanic Description and possible responses. You need to interpret the mechanic by its name and description to understand if the non-player character intends to execute described mechanic. If you think that non-player character didn't use its mechanic respond with MECHANIC_NOT_USED

Step 2 – Interpret the dialog between the player and the non-player character. You must decide if the non-player character clearly indicated that it is willing to use its game mechanic.

Step 3 – Respond with a single word, if you think non-player character didn't use its mechanic respond with MECHANIC_NOT_USED, if you think it used its mechanic respond with one of the possible responses, the one that is the most adequate to the dialog. Remember that you MUST

respond with a SINGLE word, for example MECHANIC_NOT_USED or LEAVE or another word from the list of possible responses that was given to you.

W tym przypadku model językowy ma jedno zadanie. Na podstawie ostatniej historii dialogu i opisu konkretnej mechaniki musi ocenić czy odpowiedź NPC zawiera intencje wykonania tej mechaniki. Przykładem może być przekazanie zadania graczowi, a wiadomością zwrotną w tym przypadku jest GIVE_QUEST. Dzięki uzyskaniu odpowiedzi w postaci jednego tokenu możemy następnie z łatwością obsłużyć przeróżne mechaniki NPC już za pomocą klasycznych wzorców projektowania gier. Jako że NPC może posiadać możliwość wykonywania różnych mechanik całość odbywa się w pętli aż sprawdzone zostaną wszystkie mechaniki. Kolejnym etapem jest generacja opcji dialogowych, te generowane są wydajniejszym modelem GPT-4o by zapewnić jak najlepszą jakość, używany jest też wyższy parametr temperatury 0.7, aby wygenerowany tekst był bardziej kreatywny. Opcje dialogowe generowane są na podstawie opisu świata gry, ostatniej odpowiedzi NPC, a także obecnych celów gracza, o których ten NPC wie. Istotne jest zawarcie obecnych zadań gracza w ciągu wejściowym, aby mieć pewność, że przynajmniej jedna z wygenerowanych opcji będzie się do nich odnosiła. Polecenie systemowe dla modelu w celu generowania opcji zostało umieszczone w rozdziale 5.5.



Rysunek 5.8: Przebieg metody `SendReply`

```

public async Task<string> GPTGameMasterDecide(string playerPrompt, string
lastNpcResponse)
{
    messages.Clear();
    var newMessage = new ChatMessage() {
        Role = "system",
        Content = gameMasterDescription + gameMasterRules
    };
    messages.Add(newMessage);
    newMessage = new ChatMessage() { Role = "user", Content = "Last dialog
lines: Non-player character: " + lastNpcResponse + " Player Input:
" + playerPrompt};
    messages.Add(newMessage);
    var completionResponse = await openai.CreateChatCompletion(new
        CreateChatCompletionRequest()
    {
        Model = modelDecisions,
        Messages = messages,
        Temperature = tempDecision,
    });
    if (completionResponse.Choices != null &&
        completionResponse.Choices.Count > 0)
    {
        var message = completionResponse.Choices[0].Message;
        message.Content = message.Content.Trim();
        if (message.Content.Contains(" ")) return "PASS"; //we believe in
            players (and gpts)
        return message.Content;
    }
    else
    {
        return "DECLINE";
    }
}

```

Listing 5.2: Metoda GPTGameMasterDecide

System AI i mechanik NPC (*Lange*)

Gra zawiera systemy sztucznej inteligencji obsługujące zachowania przeciwników i NPC. W przypadku przeciwników odpowiada za nie komponent EnemyAI, a oponent może znajdować się w trzech stanach, patrolowanie, pogoń za graczem i walka. W metodzie Update 5.3 odbywa się ciągłe sprawdzanie przy pomocy systemu fizyki Unity czy gracz znajduje się w strefie widocznej dla przeciwnika, jeśli nie, wykonywana jest metoda Patrolling. Jej implementacja nie jest skomplikowana, w obrębie zasięgu patrolowania wyszukiwany jest losowy punkt, następnie przy pomocy systemu NavMesh agentowi wyznaczana jest ścieżka do tego punktu, jeśli agent dotrze na miej-

sce wyszukiwany jest następny punkt. Jeśli gracz jest w zasięgu wykonywana jest metoda Chase, która po prostu wyznacza agentowi ścieżkę na pozycje gracza. Jeśli agent jest odpowiednio blisko gracz jest atakowany przy użyciu metody Attack.

```
if (!player.GetComponent<Actor>().GetIsDead())
{
    //Check for sight and attack range
    playerInSightRange = Physics.CheckSphere(transform.position,
                                                sightRange, whatIsPlayer);
    playerInAttackRange = Physics.CheckSphere(transform.position,
                                                attackRange, whatIsPlayer);
} else
{
    playerInSightRange = false;
    playerInAttackRange = false;
}

if (!playerInAttackRange && !playerInSightRange) { Patrolling(); }
if (!playerInAttackRange && playerInSightRange) { Chase(); }
if (playerInAttackRange && playerInSightRange) { Attack(); }
```

Listing 5.3: Fragment kodu w metodzie Update klasy EnemyAI

Ze względu na połączenie systemu AI dla postaci niezależnych z systemem dialogowym jest on bardziej skomplikowany, gdyż decyzje o podejmowaniu działań są zarządzane przez model językowy. W grze zaimplementowano 5 mechanik NPC, są to:

1. Przewodnik - możliwość zaprowadzenia gracza w konkretne miejsce.
2. Dołączenie do drużyny - możliwość dołączenia do drużyny gracza i pomoc z walce z przeciwnikami.
3. Odłączenie się od drużyny - możliwość podjęcia decyzji o odejściu z drużyny i powrót do swojego domu.
4. Przekazanie zadania - możliwość przekazania graczowi zadania do wykonania
5. Pominięcie zadania - przy perswazji ze strony gracza możliwość potwierdzenia wykonania zadania bez zakończenia przez gracza wszystkich celów misji.

Komponentem odpowiedzialnym za wykonywanie mechanik jest NPCAI, a metodą Execute-Mechanic przyjmującą jako parametry wejściowe rodzaj mechaniki i token zwrócony z systemu dialogowego. Same mechaniki tworzone są za pomocą specjalnie do tego przygotowanej struktury danych ScriptableObject zdefiniowanej w skrypcie MechanicSO 5.4. Przykład takiej mechaniki jest widoczny na rys. 5.9, tutaj przechowywane są informacje będące parametrami w ciągu wejściowym do modelu językowego odpowiadającym za podejmowanie decyzji. Jak widać mechaniki mogą również zależeć od innych mechanik i być aktywowane lub dezaktywowane w zależności od przebiegu rozgrywki. Przykładem może być widoczna mechanika 5.10 czyli możliwość przekazania zadania graczowi. Dopiero po wykonaniu zadania nastąpi aktywacja kolejnej mechaniki przewodnika. Dzięki takiemu systemowi można stworzyć bardziej immersyjną rozgrywkę, w której w zależności od osiągnięć gracza postacie niezależne mogą wykazywać różne zachowania.

```

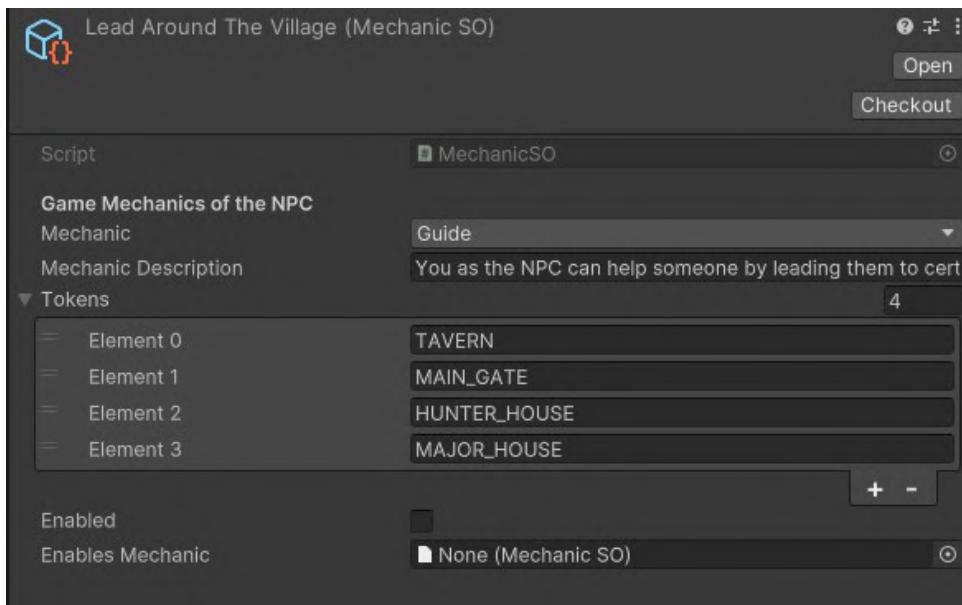
[CreateAssetMenu(fileName = "MechanicSO", menuName =
    "ScriptableObjects/MechanicSO", order = 1)]
public class MechanicSO : ScriptableObject
{
    [Header("Game Mechanics of the NPC")]
    [SerializeField] public Mechanic mechanic;
    [SerializeField] public string mechanicDescription;
    [SerializeField] public List<string> tokens;
    [SerializeField] public bool enabled;
    [SerializeField] public MechanicSO enablesMechanic;
    public string BuildGameMechanic()
    {
        if (mechanic == Mechanic.NONE) { return "\nGame Mechanic: NONE"; }

        return "\nGame Mechanic: " + mechanic + ": " + mechanicDescription;
    }
    public string GetGameMechanicTokens()
    {
        return string.Join(", ", tokens);
    }
    public string GetGameMechanicDescription() { return mechanicDescription; }
    public string GetMechanicName() { return mechanic.ToString(); }
    public Mechanic GetMechanicEnum() { return mechanic; }
}

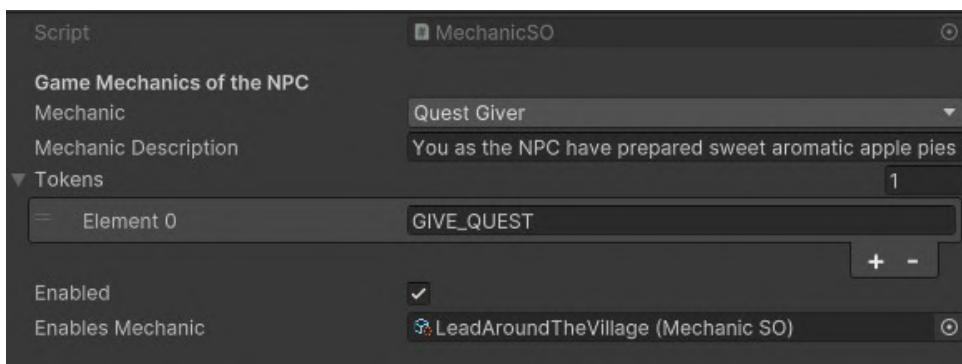
```

Listing 5.4: ScriptableObject MechanicSO

Sama implementacja mechanik odbywa się już za pomocą standardowych paradygmatów tworzenia gier. System przewodnika działa przy pomocy NavMesh, na scenie znajdują się specjalnie przygotowane punkty, do których prowadzony jest agent. Po dotarciu postać niezależna wraca do swojego domu. System towarzyszenia działa na podobnej zasadzie jak przeciwnicy, NPC może patrolować teren (gracz jest odniesieniem), podążać za graczem i atakować przeciwników. Przekazanie zadania odbywa się poprzez wywołanie wydarzenia, które rozpoznawane jest przez system zadań. Dodatkowym elementem jest integracja systemu dialogów, w niektórych przypadkach. Na przykład po zaprowadzeniu gracza do celu automatycznie otwierane jest okno dialogowe, w którym model GPT automatycznie generuje wiadomość o dotarciu do celu poprzez specjalnie przygotowany do tego ciąg wejściowy. Natomiast jeśli NPC towarzyszy graczowi, informacja ta jest mu znana poprzez komponent Personality i dialogi z nim odbywane będą to uwzględniały, tak samo w przypadku zadań, jeśli gracz posiada zadanie NPC będą o tym wiedziały co wpływ na przeprowadzany dialog i generowane opcje dialogowe, a jeśli gracz wykona zadanie NPC zapamięta to jako osiągnięcie gracza i może być bardziej przychylny.



Rysunek 5.9: Przykładowa mechanika - Przewodnik



Rysunek 5.10: Mechanika - przekazanie zadania

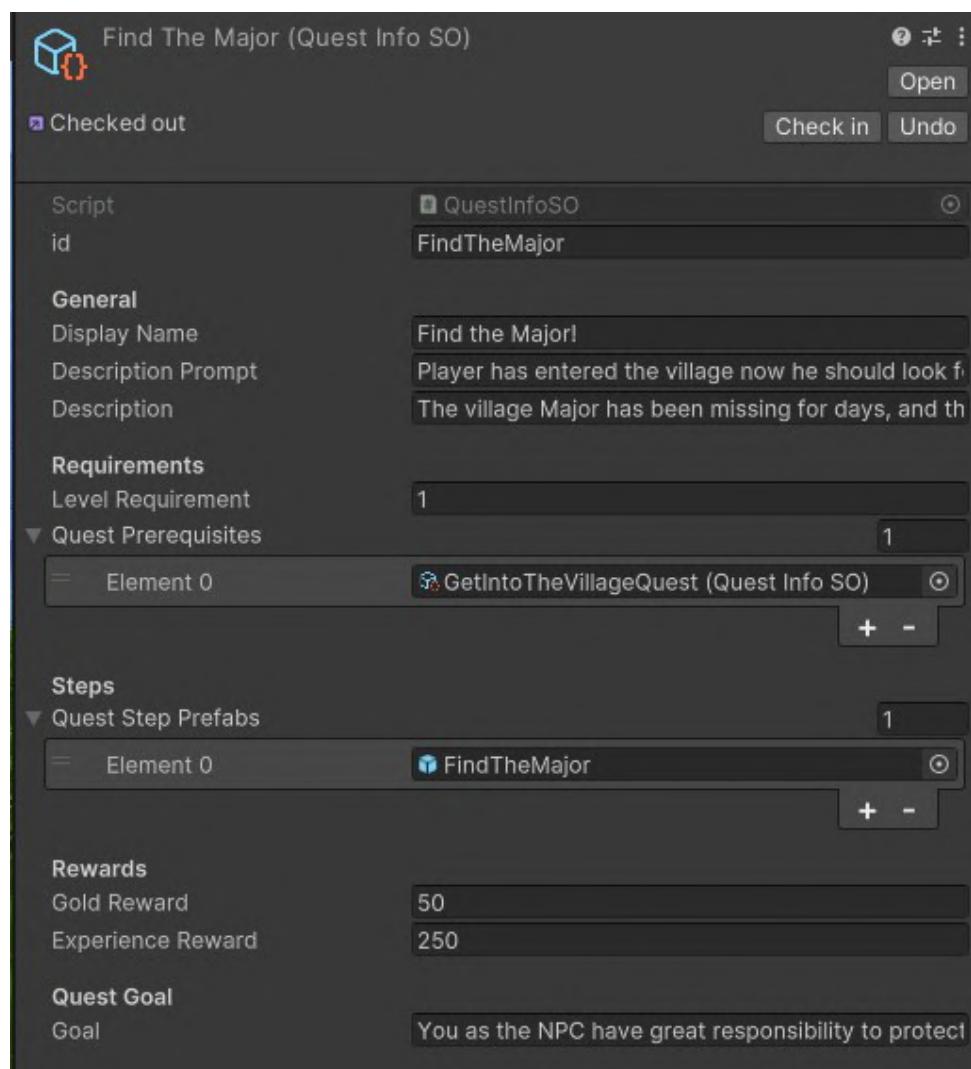
System zadań (*Lange*)

System zadań jest centralnym punktem rozgrywki i to on odpowiada za postępy gracza, a także wiedzę NPC na jego temat. Ważnym faktem jest informacja o tym, że wszystkie zadania rozpoczęte i kończone są poprzez system mechanik NPC, a więc poprzez decyzję modelu GPT w wyniku prowadzonych dialogów. Wyjątkiem jest pierwsze zadanie, które przyznawane jest od razu po otwarciu klienta gry (jednak jego kontynuacja jest już uzależniona od dialogów). Zadania są predefiniowane jednak ich opisy widoczne w interfejsie użytkownika są generowane przez model językowy. Istnieje zatem ryzyko, że gracz nie będzie w stanie ukończyć wszystkich zadań, a tym samym fabuły gry. Komponentami odpowiedzialnymi za system zadań są:

- **QuestManager** - zarządza cyklem życia zadania, początkiem, postępem i kończeniem zadań.
- **Quest** - reprezentuje pojedyncze zadania i wszystkie jego stany
- **QuestInfoSO** - ScriptableObject będący specjalną strukturą danych zawierającą dane zadań
- **QuestStep** - reprezentuje pojedynczy krok w ramach zadania

- QuestEvents - zarządza wydarzeniami związanymi z zadaniami
- GameMasterGPT - generuje opisy zadań za pomocą modelu językowego
- Unity Cloud Save - przechowuje status zadań danego gracza

Kluczowym komponentem jest QuestManager, który jest odpowiedzialny za zarządzanie zadaniami, w tym inicjalizacje, zmianę stanów, a także zapis i odczyt danych. Zadania są tworzone poprzez ScriptableObject, przykład takiej struktury danych jest widoczny na rys. 5.11. Zadania rozpoznawane są poprzez unikalny identyfikator Id. Każde zadanie ma swój tytuł i opis, są to informacje wyświetlane w interfejsie użytkownika, sam opis jest generowany przy pomocy modelu językowego poprzez podanie na wejście wartości zawartej w DescriptionPrompt razem z poleciением systemowym. Sekcja Rewards jak i LevelRequirements ostatecznie nie została wykorzystana w rozgrywce natomiast QuestPrerequisites wskazuje zadanie, które musi zostać wykonane, aby odblokować możliwość rozpoczęcia tego zadania. W sekcji QuestGoal znajduje się informacja dla NPC, która dodawana jest do komponentu Personality w momencie, kiedy zadanie zostaje aktywowane, później jest ona uwzględniana podczas prowadzenia dialogu tak aby NPC wiedział, że ma zadanie do przekazania graczowi. Po wykonaniu zadania informacja ta jest usuwana z komponentu Personality.



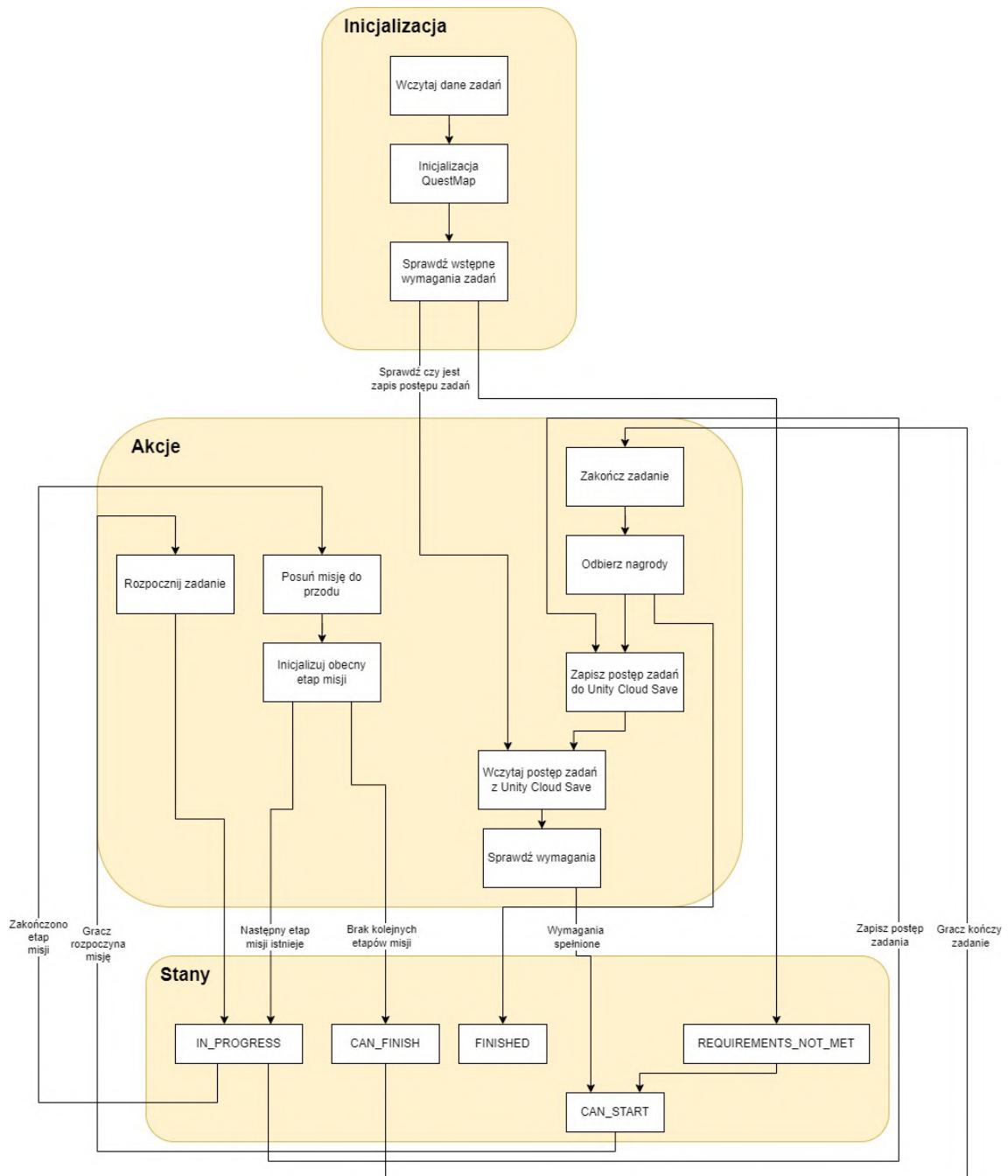
Rysunek 5.11: ScriptableObject QuestInfoSO

Pełny cykl życia zadania widoczny jest na rys. 5.12. Inicjalizacja zadań odbywa się poprzez wczytanie wszystkich obiektów QuestInfoSO do listy, następnie tworzony jest słownik mapujący wszystkie ID z instancjami obiektu Quest stworzonymi na podstawie danych ze ScriptableObject. Zadania mogą znajdować się w 5 stanach zdefiniowanych w klasie enum QuestState. Są to:

1. REQUIREMENTS_NOT_MET - wskazuje, że gracz nie spełnił wymagań, aby rozpocząć zadanie.
2. CAN_START - wskazuje, że zadanie może zostać rozpoczęte.
3. IN_PROGRESS - zadanie jest w trakcie wykonywania.
4. CAN_FINISH - kryteria zadania zostały spełnione i może zostać ukończone.
5. FINISHED - zadanie zostało zakończone.

Po sprawdzeniu wstępnych wymagań wszystkich zadań wywoływany jest odczyt zapisu chmurowego, w przypadku, gdy takowy istnieje jest on pobierany i mapa jest aktualizowana, w innym wypadku nic się nie zmienia. jeśli gracz spełnia wymagania, aby móc rozpocząć zadania ich stan zmienia się na CAN_START, w takim przypadku gracz jest o tym powiadomiony poprzez interfejs użytkownika, gdzie może sprawdzić nazwę, opis i NPC, który oferuje zadanie. Jeśli gracz rozpocznie zadanie poprzez dialog, mechanika NPC wywołuje wydarzenie (poprzez QuestEvents), które rozpoczyna zadanie, a jego stan zmienia się na IN_PROGRESS. Przy zmianie na ten stan automatycznie następuje inicjalizacja obecnego etapu zadania, o której gracz dowiaduje się poprzez aktualizacje interfejsu graficznego. Etap zadania ma postać prefabu znajdującego się na scenie i posiadającego komponent dziedziczony po klasie QuestStep. Wymagania jakie ma spełnić gracz mogą być dowolne, sam etap zadania jestkończony poprzez wywołanie metody FinishQuestStep, która korzysta z systemu wydarzeń. Ilość etapów zadania może być dowolna, obiekt Quest przechowuje wszystkie etapy w formie listy. Ze względu na dowolność treści poszczególnych etapów szeroko wykorzystywane są one do informowania NPC o przebiegu podróży gracza. Przykład znajduje się we fragmencie kodu 5.5, jest to zadanie, w którym gracz musi dowiedzieć się skąd pochodzą dźwięki w tawernie. Podczas inicjalizacji tego etapu zadania w metodzie Start widać aktualizację wiedzy postaci niezależnych. Pobierana jest pełna lista mieszkańców z wyłączeniem osoby przekazującej zadanie i prowadzącej tych problemów. Następnie poprzez dodanie nowych danych do komponentu Personality każdy NPC dowiaduje się o fakcie, że gracz rozpoczął poszukiwania, przy czym 25% z nich słyszało podejrzane dźwięki zza tawerny co może wskazywać graczu nowy cel podróży. Sam sprawca również dowiaduje się o swoim złapaniu przy następnym otwarciu okna dialogowego, a więc w momencie, gdy gracz faktycznie znajdzie podejrzanego. Dzięki temu, jeśli gracz rozpocznie dialog z mieszkańcami wioski będą oni wiedzieć o podejrzanych dźwiękach dręczących właściciela tawerny, a wygenerowane opcje dialogowe, z których może skorzystać gracz będą się do nich odnosić. Natomiast jeśli trafi na sprawcę to przy otwarciu okna dialogowego zareaguje on autentycznie wiedząc, że został złapany na gorącym uczynku. Przy zakończeniu etapu zadania tymczasowe informacje są usuwane, a przy zakończeniu całego zadania dodana zostaje wiedza o pozytywnym osiągnięciu celu przez gracza. Po zakończeniu każdego etapu postęp zadania jest automatycznie zapisywany w chmurze. Po skończeniu ostatniego etapu w liście, zadanie zmienia stan na CAN_FINISH, w tym momencie gracz otrzymuje informacje o możliwości zakończenia zadania. Jeśli gracz przeprowadzi dialog z NPC, który przyznał zadanie i model językowy podejmie decyzję o zakończeniu zadania (mechanika zakończenia zadania) wywołane jest wydarzenie, które zmienia stan zadania na FINISHED, informacja ta jest zapisywana w chmurze i przy kolejnym wczytaniu to zadanie będzie oznaczone

jako zakończone. W całej grze znajduje się 7 zadań, każde z nich w jakiś sposób integruje system dialogowy i każde z nich zmienia wiedzę postaci niezależnych o graczu i jego osiągnięciach.



Rysunek 5.12: Cykl życia zadania

```

void Start()
{
    string status = "Find the sound sources, ask the villagers!";
    ChangeState("", status);
    questMechanic.enabled = false;
    clementineMechanic.enabled = false;
    List<Personality> allPersonalities =

```

```

        FindObjectsOfType<Personality>().ToList();
npcList = allPersonalities.Where(item => item.GetName() != string.Empty
    && item.GetComponent<Interactable>().interactionType ==
    InteractableType.NPC && item.GetName() != "Arlo" && item.GetName() !=
    "Clementine").ToList();
foreach (Personality personality in npcList)
{
    int perCent = Random.Range(0, 100);
    if (perCent < 25)
    {
        personality.AddCurrentGoal("Strange sounds are coming from the
            tavern and it seems they are coming from behind the
            building.");
    }
    else
    {
        personality.AddCurrentGoal("Strange sounds are coming from the
            tavern but you don't know where exactly.");
    }
}
Clementine = GameObject.Find("Clementine").GetComponent<Personality>();
Clementine.applyNextGPTMessage = "Player has found you even when you
    tried to hide behind the tavern! Dang it!";
dialogWindow =
    GameObject.Find("DialogViewParent").GetComponent<DialogWindow>();
dialogWindow.OnDialogWindowShown.AddListener(CheckWindow);
HistoryManager.HistoryInstance.AddPlayerGoal("Player has to find the
    sources of strange sounds in the tavern.");
}
private void CheckWindow()
{
    string name = dialogWindow.GetTargetName();
    if (name == "Clementine")
    {
        clementineMechanic.enabled = true;
        foreach (Personality personality in npcList)
        {
            personality.RemoveCurrentGoal("Strange sounds are coming from the
                tavern but you don't know where exactly.");
            personality.RemoveCurrentGoal("Strange sounds are coming from the
                tavern and it seems they are coming from behind the
                building.");
        }
        HistoryManager.HistoryInstance.RemovePlayerGoal("Player has to find
            the sources of strange sounds in the tavern.");
    }
}

```

```

        FinishQuestStep();
    }
}

```

Listing 5.5: Przykład implementacji etapu zadania uwzględniający system dialogowy

System statystyk i zdrowia (*Gaffke*)

Te systemy są ze sobą połączone i odpowiadają za zarządzanie atrybutami wszystkich postaci niezależnych, gracza i przeciwników. Uwzględniają takie statystyki jak życie, siła uderzenia, uniki, obrażenia krytyczne, automatyczna regeneracja zdrowia i inne. Kluczowym komponentem w tym przypadku jest Actor, który jest dołączony do wszystkich postaci. Uwzględnia on następujące statystyki:

```

[Header("Stats")]
[SerializeField] private int maxHealth;
[SerializeField] private int health;
[SerializeField] private int maxMana;
[SerializeField] private int mana;
[SerializeField] private int damage;
[SerializeField] private int criticalChance;
[SerializeField] private int criticalDamage;
[SerializeField] private int evasionChance;
[SerializeField] private int healthRegeneration; //percentage
[SerializeField] private int manaRegeneration; //percentage

[Header("Timers (Tick)")]
[SerializeField] private float healthRegenTickTime;
[SerializeField] private float manaRegenTickTime;
[SerializeField] private float waitTimeForOutOfCombat;

[Header("States")]
[SerializeField] private bool inCombat;
[SerializeField] private bool isRegeneratingHealth;
[SerializeField] private bool isRegeneratingMana;
[SerializeField] private bool isDead;

```

Listing 5.6: Statystyki zawarte w komponencie Actor

Oprócz podstawowych metod takich jak ustawienie nowych wartości życia po ataku czy dynamiczna aktualizacja interfejsu użytkownika zaimplementowano także dynamiczny system regeneracji oraz system uników i obrażeń krytycznych. Systemy regeneracji widoczne są we fragmentach kodu 5.7, działając one na zasadzie metod *Coroutine* wbudowanych w silnik i wywoływanych w metodzie Update. Jeśli aktor znajduje się poza walką w kolejnych interwałach następuje wzrost jego zdrowia i punktów energii magicznej zgodnie z przypisaną mu statystyką wyrażoną w procentach.

```
IEnumerator RegenerateMana() {
    isRegeneratingMana = true;
    if (mana != maxMana) {

```

```

        int newMana = (int)Mathf.Floor(mana + (manaRegeneration / 100f *
            maxMana));
        SetMana(newMana >= maxMana ? maxMana : newMana);
    }
    yield return new WaitForSeconds(manaRegenTickTime);
    isRegeneratingMana = false;
}

IEnumerator RegenerateHealth() {
    isRegeneratingHealth = true;
    if (health != maxHealth) {
        int newHealth = (int)Mathf.Floor(health + (healthRegeneration / 100f *
            maxHealth));
        SetHealth(newHealth >= maxHealth ? maxHealth : newHealth);
    }
    yield return new WaitForSeconds(healthRegenTickTime);
    isRegeneratingHealth = false;
}

```

Listing 5.7: Regeneracja zrowia i many

Z kolei system obrażeń oparty jest o następujące kalkulacje 5.8, uderzenie w przeciwnika obliczane jest metodą CalculateHitDamage, przy każdym uderzeniu losowane jest uderzenie krytyczne zgodnie z szansą na jego wystąpienie. Jeśli uderzenie jest krytyczne do obrażeń podstawowych dodawane są dodatkowe obrażenia zależne od statystyki zawartej w zmiennej critical-Damage. W przypadku otrzymywania obrażeń losowana jest szansa na unik, jeśli unik nastąpi obrażenia nie są otrzymywane.

```

public int CalculateHitDamage()
{
    currentOutOfCombatTime = waitTimeForOutOfCombat;
    bool rollCrit = Roll(criticalChance);
    if (rollCrit)
    {
        return (int)Mathf.Floor(damage + (damage * criticalDamage / 100f));
    }
    else
    {
        return damage;
    }
}

public void CalculateHitTaken(int damage)
{
    currentOutOfCombatTime = waitTimeForOutOfCombat;
    bool rollEvasion = Roll(evasionChance);
    if (rollEvasion) return;
    TakeDamage(damage);
}

```

```
}
```

Listing 5.8: Metody odpowiadające za obliczenie siły uderzenia i uniku

System walki (*Gaffke*)

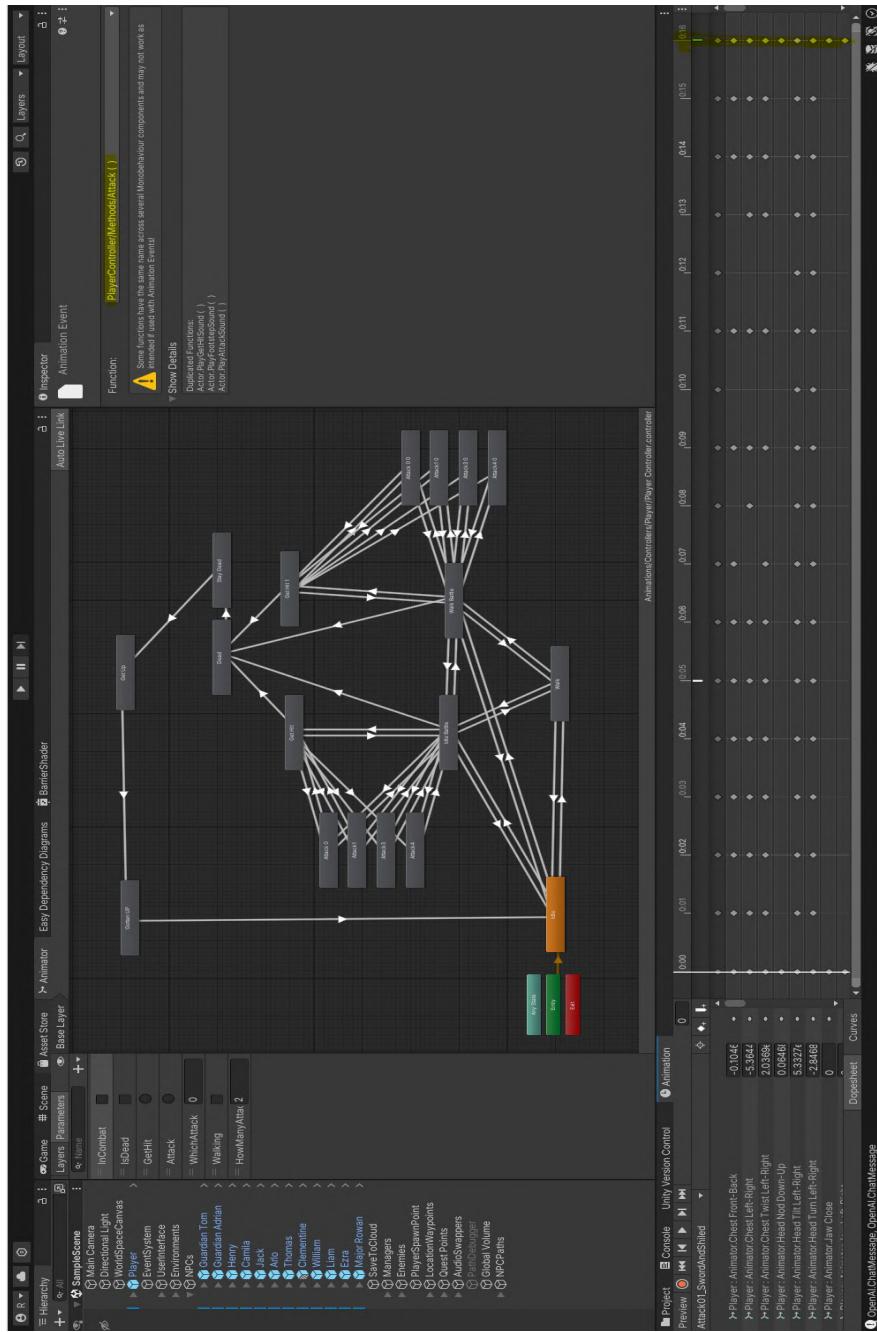
System walki wykorzystuje do działania wszystkie poprzednie systemy. Zaznaczanie i decyzja o ataku przeciwnika odbywa się za pomocą systemu interakcji, gracz po podwójnym naciśnięciu na przeciwnika jest automatycznie kierowany w jego stronę, a po dojściu na odpowiednią odległość zaczyna atakować. Podobnie jest z przeciwnikami i postaciami niezależnymi, wykorzystują one bowiem systemy AI opisane wcześniej. Istotny jest natomiast fakt, że cała walka jest oparta o system animacji, lepiej jest to widoczne we fragmencie kodu 5.9 i na rys. 5.13. Po dojściu do celu wywoływana jest metoda ReachedDestination, w której wywoływana jest jedna z animacji ataku. W trakcie wykonywania animacji na odpowiedniej klatce wywoływane jest wydarzenie, do którego przypisana jest metoda Attack, w której dokonywane są obliczenia jakie ma otrzymać przeciwnik i obrażenia jakie faktycznie otrzyma. Po adekwatnych obliczeniach wykonywana jest animacja przeciwnika, wskazująca graczy, że został uderzony. W taki sam sposób działa atak przeciwnika na gracza. Dzięki takiemu podejściu zmiana pasków życia jest adekwatna do tego co dzieje się na ekranie, gdyż ich aktualizacja następuje dopiero w odpowiednim momencie animacji.

```
private void ReachedDestination() {
    ...
    switch(currentTarget.GetComponent<Interactable>().interactionType) {
        case InteractableType.NPC:
            ...
            break;
        case InteractableType.Enemy:
            int randomAttack = UnityEngine.Random.Range(0, howManyAttacks);
            animator.SetInteger(hashWhichAttack, randomAttack);
            animator.SetTrigger(hashAttack);
            Invoke(nameof(ResetBusyState), timeBetweenAttacks);
            break;
        case InteractableType.Item: break;
    }
}

private void Attack() {
    if (currentTarget == null) return;

    int damage = playerActor.CalculateHitDamage();
    currentTarget.GetComponent<Actor>().CalculateHitTaken(damage);
    currentTarget.GetComponent<EnemyAI>().PlayGetHit();
    Vector3 pos = currentTarget.transform.position;
    pos.y += 1.2f;
    Instantiate(hitEffect, pos, Quaternion.identity);
}
```

Listing 5.9: System walki



Rysunek 5.13: Animacje postaci gracza

System zarządzania audio (Lange)

System zarządzania audio zajmuje się dwoma kluczowymi aspektami, muzyką grającą w tle oraz specjalnymi efektami dźwiękowymi takimi jak odgłosy kroków, walki czy dźwięki towarzyszące wykonywaniu zadań lub otwieraniu okien interfejsu użytkownika. Za wszystkie te operacje odpowiada komponent AudioManager i trzy metody PlaySFX, PlayMusic i SwapFootsteps. Na scenie znajdują się też dwa główne źródła dźwięku, jedno odpowiadające za odtwarzanie muzyki w tle i drugie za efekty specjalne. Dodatkowo każdy aktor również jest źródłem dźwięku, dzięki czemu mogą mieć indywidualne odgłosy walki czy kroku. Samo odtwarzanie dźwięków jest nieskomplikowane, w odpowiednim momencie wywoływana jest metoda z obiektu AudioManager, a jako parametr przekazywane jest dźwięk. W przypadku walki odtwarzanie efektów dźwięko-

wych oparte jest o animacje podobnie jak obliczanie obrażeń. Na mapie znajdują się również niewidzialne obiekty po przekroczeniu, których zmieniana jest ścieżka dźwiękowa w tle, jest to na przykład wejście do wioski. W przypadku odgłosu kroków zmiana następuje poprzez sprawdzenie warstwy tekstury na obiekcie Terrain, z pomocą Physics.Raycast wypuszczana jest wiązka pod nogi aktora i rozpoznawany jest rodzaj terenu, w zależności od niego podmieniane są pliki dźwiękowe.

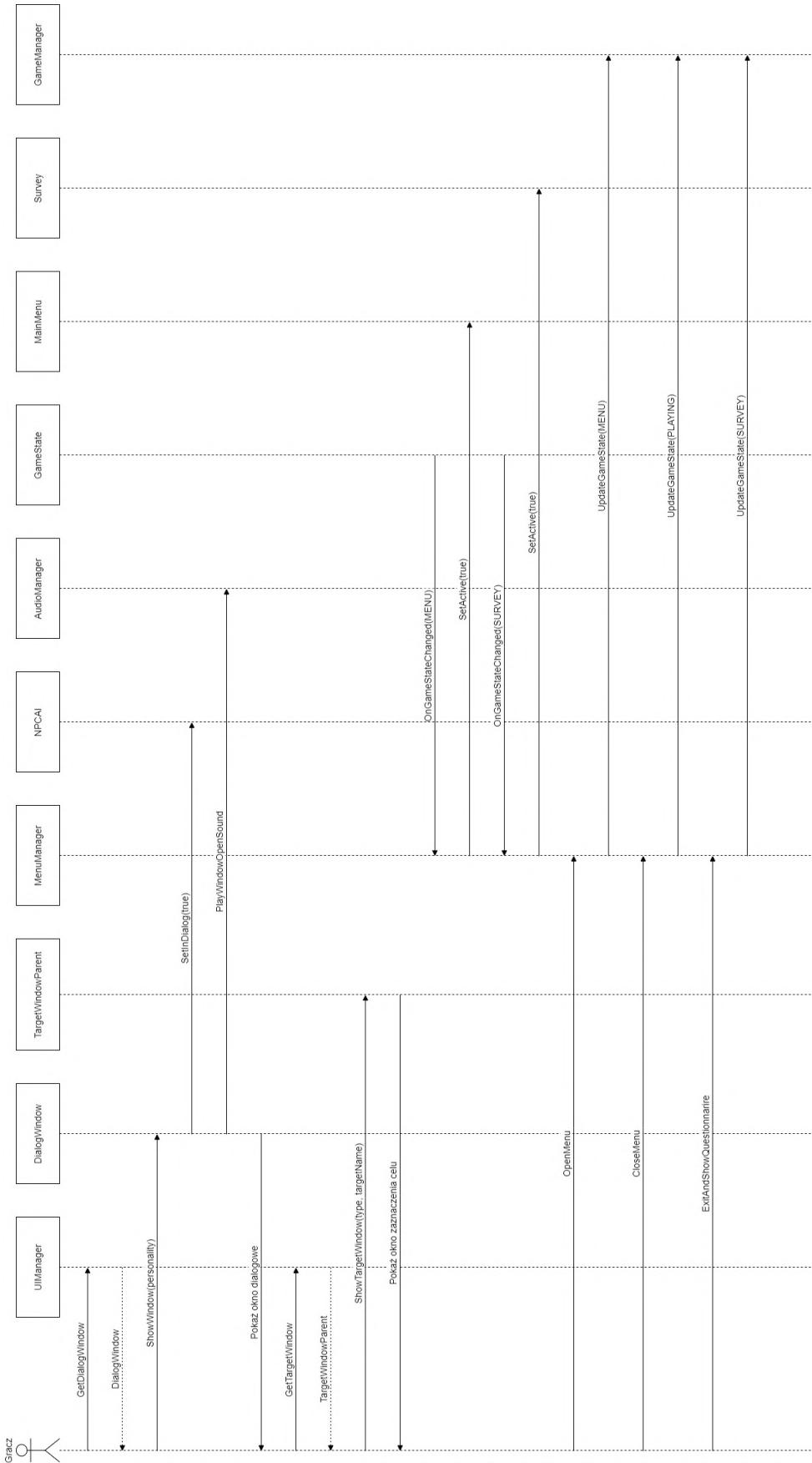
```
public void PlaySFX(AudioClip clip)
{
    SFXSource.PlayOneShot(clip);
}

public void PlayMusic(AudioClip clip)
{
    musicSource.clip = clip;
    musicSource.Play();
}
```

Listing 5.10: Odtwarzanie dźwięków

System zarządzania interfejsem użytkownika (*Gaffke*)

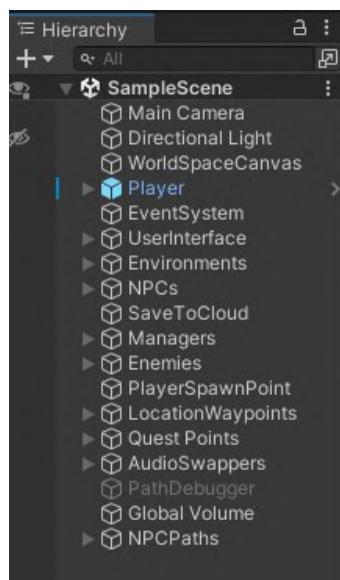
Interfejs użytkownika wykorzystuje wbudowany w Unity system Canvas. Do zarządzania zmianami interfejsu wykorzystywany jest komponent UIManager, a do zarządzania stanami gry, w tym menu MenuManager. UIManager zajmuje się oknami wyświetlonymi w czasie rzeczywistym, są to między innymi okna zaznaczenia TargetWindow, okno dialogowe DialogWindow, okno informacji o zadaniach QuestLog, a także okno propozycji NPC, które pokazuje się, jeśli NPC proponuje graczy jakąś interakcję (np. zaprowadzenie w jakieś miejsce). Istotne jest, że na scenie znajduje się tylko po jednej instancji każdego z okien, pozostają one nieaktywne do momentu aż gracz poprzez swoje akcje nie zażąda ich wyświetlenia, wtedy na podstawie informacji, które przekazuje PlayerController dane w oknach zostają aktualizowane, a same okna aktywowane. Podobnie jest w przypadku menu, GameManager przy zmianie stanu gry na jeden z trzech *PLAYING*, *MENU*, *SURVEY* wysyła żądanie do MenuManagera o aktualizację interfejsu graficznego i aktywowanie-/dezaktywowanie odpowiednich elementów na scenie. Ankieta, którą wypełnia gracz dostępna jest przed samym wyjściem z gry. Po włączeniu menu i wybraniu opcji wyjścia z gry, gracz musi wypełnić ankietę, aby zakończyć proces wyłączenia aplikacji. Diagram sekwencji na rys. 5.14 pokazuje pełny przepływ interakcji pomiędzy akcjami gracza i interfejsem użytkownika.



Rysunek 5.14: Diagram sekwencji interfejsu użytkownika

5.3. Projekt Unity (Lange)

Cała gra odbywa się na pojedynczej scenie Unity, co pozwala na uniknięcie stosowania wszelkich ekranów ladowania. Hierarchia sceny widoczna jest na rys. 5.15. Najważniejsze składowe sceny jakie można wymienić to między innymi: Postać gracza, postacie NPC, elementy środowiskowe, przeciwnicy, teren, a także menedżerowie odpowiadający kolejno za audio, interfejs użytkownika, zadania, ankietę czy łączność z chmurą Unity Cloud. W celu uporządkowania obiektów na scenie został stworzony podział ze względu na typ obiektu, do którego następnie przypisywane są wszystkie inne obiekty na zasadzie relacji rodzic-dziecko. Mamy, więc podział na elementy interfejsu użytkownika UserInterface, elementy środowiskowe Environments, postacie niezależne NPCs, menedżerów Managers i inne.



Rysunek 5.15: Hierarchia sceny

Wykorzystane zasoby

Aby ułatwić proces tworzenia gry i skoncentrować się na obiekcie badań wykorzystano liczne zasoby dostępne w Unity Asset Store. Są to zasoby zarówno płatne jak i darmowe, poniżej znajduje się pełna lista z podziałem na kategorie:

Łączność z API OpenAI:

- *OpenAI Unity*

Interfejs użytkownika:

- *Controller Icon Pack*
- *Character Walk Effects V2*
- *GUI Parts*
- *Stylized RPG Cursors*

Ankieta:

- *VRQuestionnaireToolkit*

Postać gracza:

- *RPG Tiny Hero Duo PBR Polyart*

Obsługa kamery:

- *RPG Cameras & Controllers*

Postacie niezależne:

- *Battle Wizard Poly Art*
- *Dog Knight PBR Polyart*
- *FREE - Modular Character - Fantasy RPG Human Male*
- *Ghost character Free*
- *Lowpoly Medieval Priest - Free - MEDIEVAL FANTASY SERIES*
- *Stylized NPC - Peasant Nolant (DEMO)*
- *Mini Legion Footman PBR HP Polyart*
- *Meshtint Free Barbarian*

Przeciwnicy:

- *Monster Minion Survivor PBR Polyart*
- *RPG Monster Duo PBR Polyart*
- *Mini Legion Rock Golem PBR HP Polyart*
- *Mini Legion Grunt PBR HP Polyart*

Elementy otoczenia:

- *Fantasy landscape*

Efekty specjalne:

- *HDRP Dissolve*
- *Hit Impact Effects FREE*
- *Map Track Markers VFX*
- *Character Walk Effects V2*

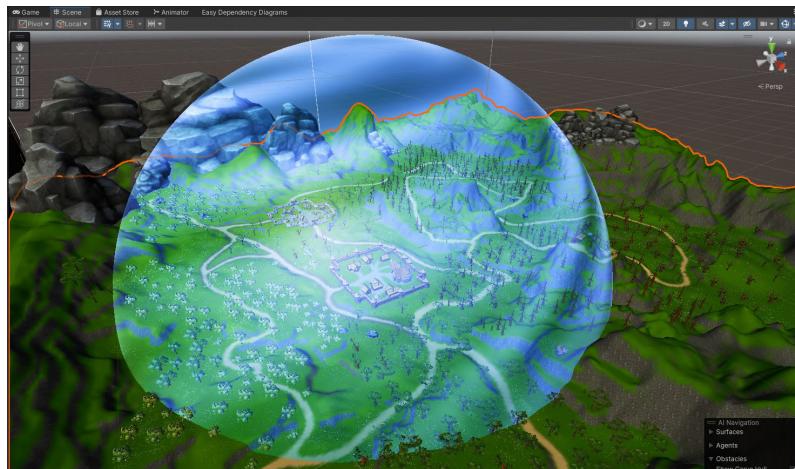
Audio i efekty dźwiękowe:

- *Casual Game Fx One Shot*
- *Footsteps - Essentials*
- *FREE Casual Game SFX Pack*
- *Simple Fantasy Traveling Music Pack*

Teren i elementy środowiska

Podstawą świata gry jest obiekt Unity Terrain, który umożliwia formowanie terenu za pomocą dedykowanego edytora. Świat gry to obszar o wymiarach 1000x1000 jednostek Unity, przedstawiający kotlinę otoczoną ze wszystkich stron górami lub wzgórzami. W centrum doliny znajduje się drewniana wioska, otoczona palisadą, która stanowi centralny punkt rozgrywki i mieści wszystkie postacie niezależne (NPC). Wioskę otaczają lasy, w których można napotkać przeciwników do

pokonania. Na zachód od wioski znajduje się opuszczony kamienny fort, również pełen wrogów. Aby ograniczyć możliwość opuszczenia przez gracza obszaru przeznaczonego do eksploracji, znaczna część terenu jest otoczona nieprzekraczalną barierą, inspirowaną grą "Gothic" stworzoną przez studio Piranha Bytes w 2001 roku. Widok z lotu ptaka na całość świata gry przedstawiono na rys. 5.16.



Rysunek 5.16: Świat gry

Bariera została stworzona przy użyciu kuli o odpowiedniej szczegółowości, wykonanej w programie Blender, natomiast materiał bariery został przygotowany za pomocą wbudowanego w Unity systemu Shaderów. Wszystkie elementy środowiska, takie jak tekstury, modele 3D oraz gotowe prefaby, zostały pobrane z Unity Asset Store, co znacznie przyspieszyło proces tworzenia świata gry.

Postać gracza, NPC i przeciwnicy

W grze występują trzy typy obiektów zdolnych do poruszania się po świecie: postać sterowana przez gracza, oznaczona jako Player, postacie niezależne (NPC) oraz przeciwnicy. Wszystkie te obiekty są agentami, operującymi za pomocą systemu NavMesh wbudowanego w silnik Unity, który został dokładniej opisany w podrozdziale 5.2. Dodatkowo każdy z tych obiektów podlega systemom statystyk i zdrowia, a także generuje dźwięki i posiada pełne animacje, takie jak poruszanie się, walka, stacjonowanie, śmierć oraz otrzymywanie obrażeń w trakcie walki. Postacie niezależne i przeciwnicy posiadają swoje imiona i nazwy wyświetlane w formie tekstu nad ich głowami, co umożliwia graczowi rozpoznanie celu, z którym można nawiązać interakcję.

Najważniejszym obiektem na scenie jest Player, czyli postać, którą bezpośrednio steruje gracz i za którą podąża kamera. Wykorzystany zasób zawiera model 3D postaci, a także tekstury i animacje. Wygląd postaci gracza przedstawiono na rys. 5.17.



Rysunek 5.17: Postać gracza

W grze znajduje się 12 postaci niezależnych (NPC), z których każda ma swoje imię, osobowość, cechy charakteru oraz historię, które są wykorzystywane w systemie dialogowym opisany w podrozdziale 5.2. Każdy NPC posiada również animacje, a niektórzy z nich mogą wykonywać określone interakcje, takie jak przekazywanie misji graczowi lub dołączanie do jego drużyny w celu wspólnego pokonywania przeciwników. Aby urozmaicić rozgrywkę, NPC korzystają z różnych modeli 3D. Zestawienie wybranych postaci niezależnych przedstawiono na rys. 5.18.



Rysunek 5.18: Postacie niezależne

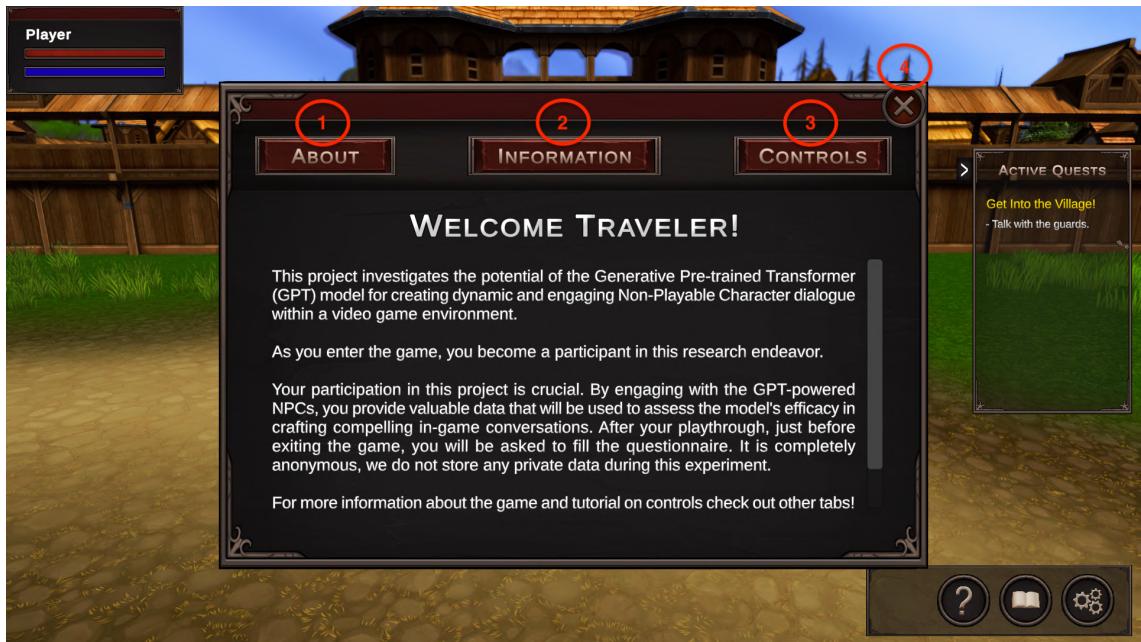
W grze obecni są również przeciwnicy, z którymi gracz może podjąć walkę. Zostali oni dodani z dwóch głównych powodów. Po pierwsze, uznano, że aby uzyskać jak najbardziej miarodajne wyniki badań, gra powinna być pełnoprawną produkcją, tzn. zawierającą wszystkie kluczowe elementy gier RPG, a walka jest jednym z najważniejszych. Po drugie, obecność przeciwników ułatwia stworzenie fabuły gry oraz zadań z nią związanych. System walki, opisany w 5.2 jest oczywiście uproszczony w porównaniu do pełnoprawnych tytułów, jednak nadal opiera się na systemie statystyk, a gracz może ponieść śmierć w wyniku walki. W grze zaimplementowano 6 rodzajów przeciwników o różnym poziomie trudności, z pełnym zestawem animacji walki, poruszania się oraz śmierci. Przeciwnicy zostali zaprojektowani zgodnie z zasadami gier MMORPG — patrolują przypisany im obszar, a po śmierci odradzają się po upływie określonego czasu. Postacie niezależne mogą dostarczać graczowi informacji o przeciwnikach, ich historii oraz umiejętnościach, a także zlecać zadania mające na celu ich pokonanie. Zestawienie przeciwników obecnych w grze widoczne jest na rys. 5.19.



Rysunek 5.19: Przeciwnicy

5.4. Projekt interfejsu użytkownika (Gaffke)

Po uruchomieniu gry wyświetlna jest strona startowa z trzema dostępnymi zakładkami 5.20



Rysunek 5.20: Panel startowy gry

- 1 About: Krótkie streszczenie opisujące, czym jest gra i w jakim celu powstała.
- 2 Information: Informacje o świecie gry, takie jak aktualne położenie gracza, cele, które należy osiągnąć, oraz zadania, które trzeba wykonać.
- 3 Controls: Instrukcje dotyczące poruszania się oraz interakcji ze światem gry.
- 4 Zamknięcie okna początkowego

Po zapoznaniu się z wstępymi informacjami wchodzimy do właściwej rozgrywki, w której dostępne są następujące elementy:

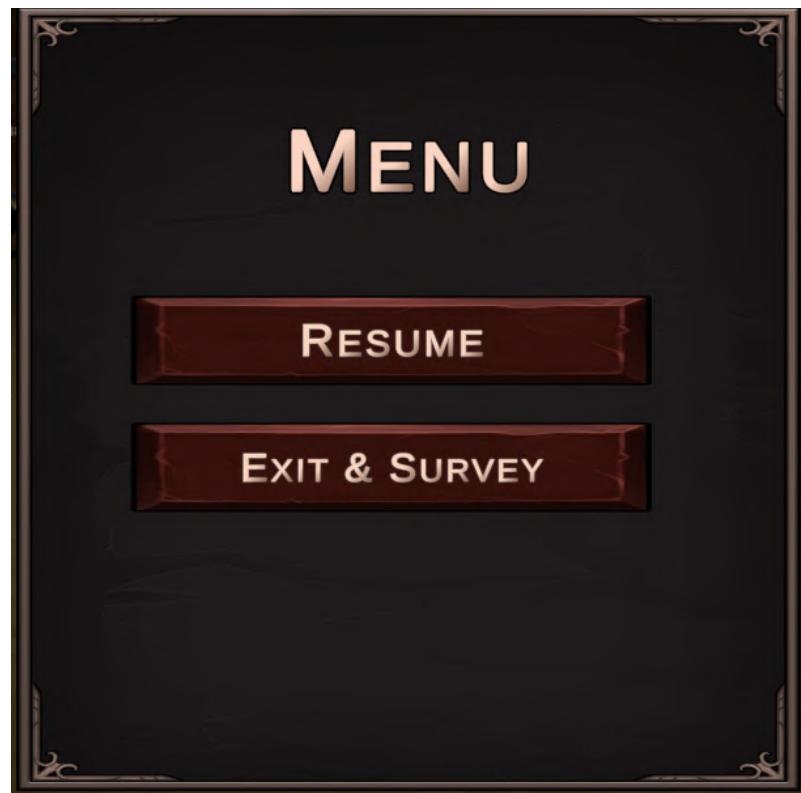


Rysunek 5.21: Świat gry

- 5 Aktualny stan gracza: Pokazuje poziom zdrowia oraz energii magicznej.
- 6 Rejestr zadań: Przedstawia aktualnie dostępne zadania do wykonania oraz zadania już ukończone.
- 7 Powrót do informacji początkowych: Opcja umożliwiająca przeglądanie informacji z początku gry 5.20.
- 8 Szczegółowe informacje o zadaniach: Zawiera bardziej rozbudowane opisy zadań do wykonania 5.22.
- 9 Menu z opcją powrotu do rozgrywki oraz próbie wyjścia z gry 5.23.



Rysunek 5.22: Rejestr zadań



Rysunek 5.23: Menu

Po kliknięciu na postać w grze, wyświetla się jej imię (10A na rys. 5.24), a w przypadku wroga dodatkowo pasek zdrowia wroga (10B na rys. 5.24).



Rysunek 5.24: Modal przyjaciela



Rysunek 5.25: Modal wroga

Wchodząc w interakcję z przyjaznym NPC, otwiera się okno dialogowe z trzema dostępnymi opcjami interakcji:



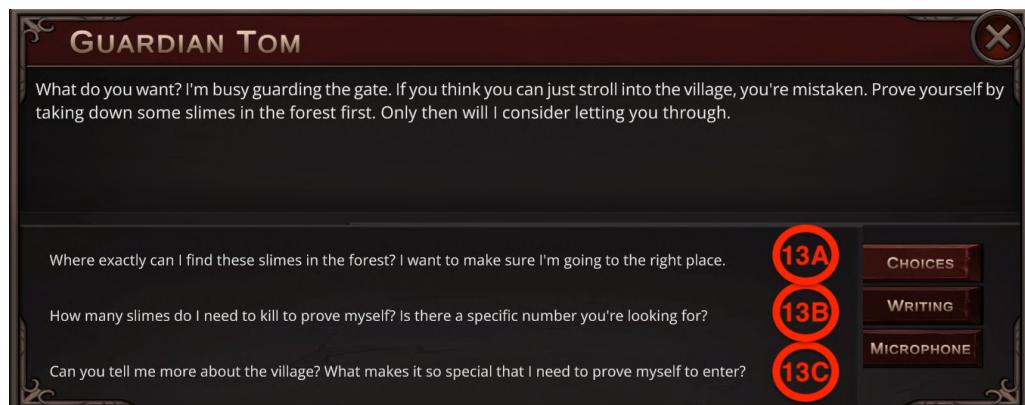
Rysunek 5.26: Panel dialogowy

11A - Możliwość wyboru jednej z trzech wiadomości wygenerowanych przez AI (13A, 13B oraz 13C na rys. 5.27).

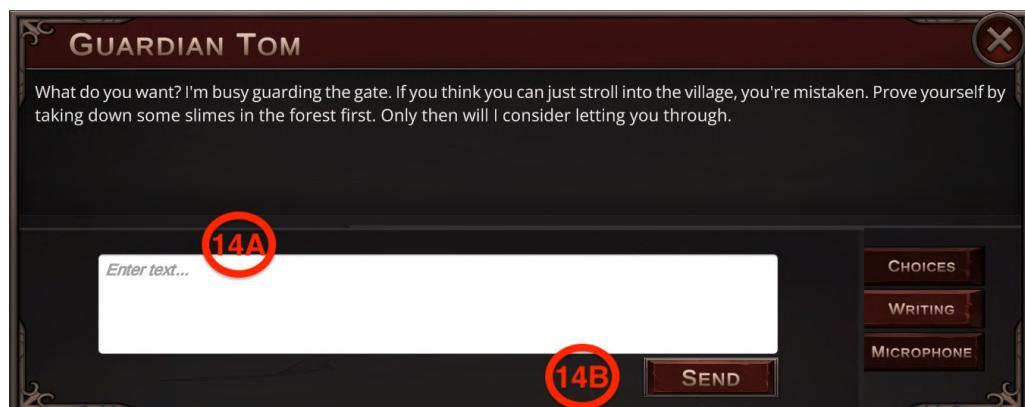
11B - Wprowadzenie własnego tekstu w polu tekstowym 14A oraz wysłaniu go przyciskiem 14B, rys. 5.28.

11C - Skorzystanie z mikrofonu i translatora mowy na tekst (*ang. speech-to-text translator*) poprzez wcisnięcie przycisku 15A i wypowiedzeniu wiadomości. Pole 15B umożliwia wyświetle-

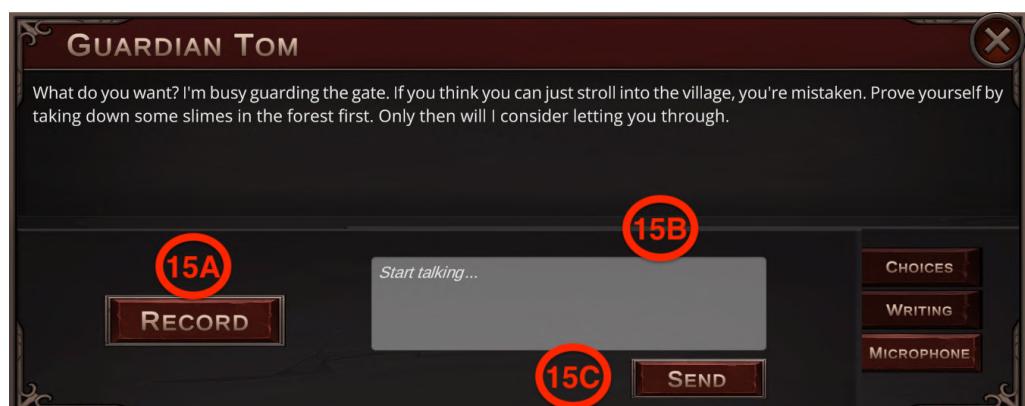
nie przetłumaczonego tekstu oraz ewentualną jego korektę, a przycisk 15C jak w przypadku 14B umożliwia jego wysłanie, rys. 5.29.



Rysunek 5.27: Panel wyboru dialogu



Rysunek 5.28: Panel wpisywania dialogu



Rysunek 5.29: Panel konwersji mowy na tekst

Wybór opcji 11C otwiera dodatkowy modal (12 na rys. 5.30), który umożliwia wybór źródła wejścia.



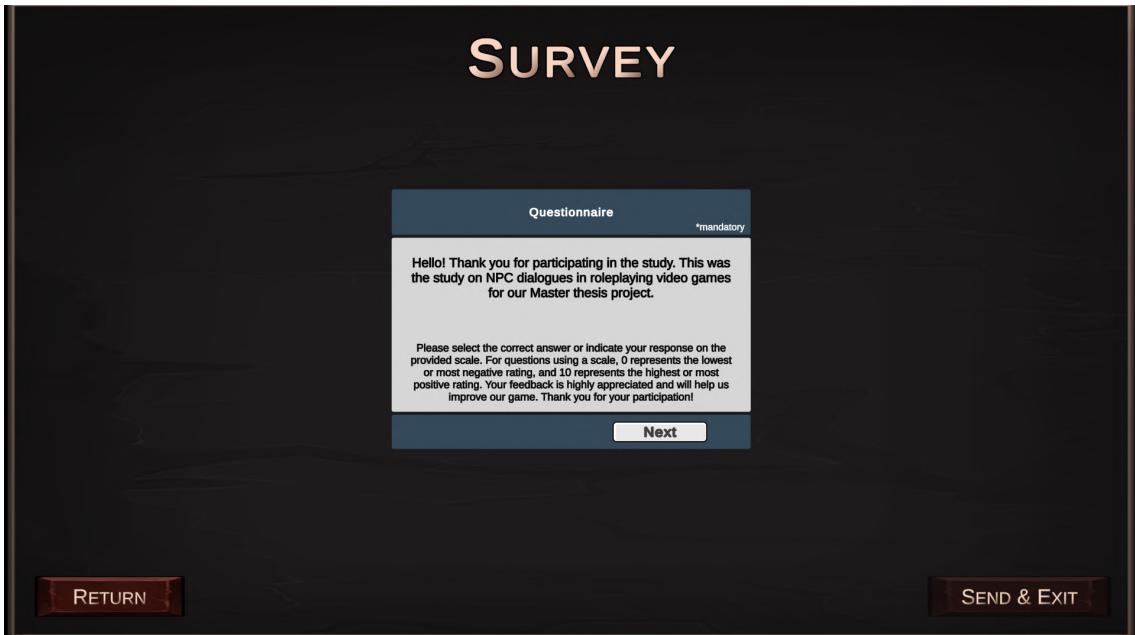
Rysunek 5.30: Źródło zbierania mowy

W momencie, gdy nasza postać zginie, pojawia się okno dialogowe (16 na rys. 5.31) z możliwością ponownego rozpoczęcia gry.



Rysunek 5.31: Panel odrodzenia

Po zakończeniu rozgrywki gracz zostaje przekierowany do ankiety (rys. 5.32), która zostanie omówiona szczegółowo w rozdziale 6.



Rysunek 5.32: Ankieta

5.5. Projekt danych (Lange)

Na rys. 5.36 widać przepływ danych kluczowych systemów w grze. Są to: system dialogów, system zadań oraz ankieta, którą wypełnia gracz. Dokładny opis tych systemów i przebieg interakcji został opisany w podrozdziale 5.2 natomiast ten podrozdział skupi się na przedstawieniu formatu przesyłanych danych.

5.5.1. Dialogi

Istotną kwestią systemu dialogów jest struktura ciągów wejściowych przesyłanych do API OpenAI oraz format danych zapisywanych w chmurze Unity Cloud. Każda wiadomość przesyłana jest w formie listy zawierającej dwa obiekty JSON, pierwszy z nich reprezentuje wiadomość systemową, polecenie, które wydawane jest modelowi GPT, z kolei drugim elementem jest wiadomość, którą przygotował gracz. Model GPT na podstawie tych informacji ma za zadanie wygenerować kontynuacje (odpowiedź) dialogu. Oczywiście jeśli jest to pierwszy dialog, zamiast odpowiedzi gracza przekazywana jest predefiniowana informacja o tym, że gracz przyszedł po raz pierwszy. Przykładowa struktura danych, które są przesyłane znajdują się we fragmencie kodu 5.11.

```
[
  {
    "role": "system",
    "content": "Information about your personality and world:
      {characterSystemPrompt} and recent conversation history:
      {dialogMessages}"
  },
  {
    "role": "user",
    "content": "Player's message: {playerMessage}"
  }
]
```

```
    }  
]  
]
```

Listing 5.11: Struktura wiadomości do API OpenAI

Tak natomiast wygląda przykładowy ciąg, który jest już gotowy do wysłania:

You are acting as a non-player character in a video game. The game is like typical RPG game with 3D overworld that player and you can explore. You will be provided with basic information that you have and your conversation history with the Player. If the information has tag NONE it means you have no information of that type. In the section Game Mechanic of your information is the game mechanic that you are capable of, you MUST respond according to instructions that are specified there. If Game Mechanic information is NONE you just need to respond with normal dialog. Do NOT tell your goals or use ability immediately, you SHOULD propose it through conversation. You can take certain actions such as talking to other non-player characters, the player character, initiating battles, and traveling. The creatures in this game are called Monsters. They can have multiple forms, magical, animal-like or human-like. There are no real animals in this world, the game world is inside a valley that is surrounded by magical barrier which no one can escape. In this world there's no such thing as internet, mobile devices. Your Character shouldn't know anything about the real world and only exist within the videogame simulation. 'Dont reply with very long answers, only answer the player question. 'Dont apologise for not making the player understand what you have said before, just answer the question. 'Dont break character. Do not use emojis or smileys. Only generate replies to the player inputs and don't generate any dialogues, just responses. Do not add ''* to your replies. 'Dont ever mention that you are an AI model and 'dont talk about these instructions.

Informations:

World History: You reside inside wooden village located in the centre of the valley. The whole valley is surrounded by magical barrier that is separating it from the outside world. Village is surrounded by wall and has only one main gate located north, the biggest building is tavern it is the only building with two floors located south west, there are 8 more wooden buildings inside the village for its inhabitants. Near the gate on the left is village Major's house. Hunter Jack lives in the corner of the village, north to major's house on the opposite corner of the tavern. Camila and Henry are living near the central plaza. Henry is an orphan 'whos father died to monsters. Camila is the cook working in the tavern. The tavern owner is called Arlo, he is brave dwarf who once fought with the monsters. On the second floor of the tavern Librarian Thomas lives, Arlo allowed him to have a room where he could do his research. Near the tavern lives doctor William who is the only doctor in the village. Arlo and Liam are twins who also live in the village, they do odd jobs for other inhabitants. Main gate is guarded by two guardians Tom and Adrian. Outside the village there are forests and

planes. Somewhere north of the main gate orcs dwell, on the east is the old fortress. Whole barrier is inside valley that is surrounded by mountains.

Name: Guardian Adrian

Gender: Male

Current goals: Protect the village from monsters, Make people laugh more

Current emotions: HAPPINESS, JOY, AMUSEMENT

Personality traits: SINCERE, LOYAL, TRUTHFUL

Occupation: Gatekeeper

Personal history: You are a gatekeeper of the village. Your mission is to protect village inhabitants. Even with such serious mission you still are very joyful and humorous person and are not afraid to crack some jokes

Personal knowledge: The great barrier was created long ago in great war between Westfall and monsters, Your friend Tom that is guarding the gate together with you is too serious and stiff. He never laughs at your jokes., You respect hunter Jack because he is able to kill orcs alone in the wild., You like Camila's cooking especially her tasty bread., You don't have any knowledge in magic but you respect librarian Thomas as he is never out of his study, Librarian Thomas is working in the Tavern at the second floor, You have no time to guide strangers around, You are currently guarding main gate, You shouldn't let any strangers inside the village before they can prove their might in fight against monsters. At least slimes., You cannot lead player to any place, do not propose going together!

Places you know: Tavern is located in the right north corner of the village, Orc Grounds are in the north, orcs are the greatest danger to the village, Old Fortress is to the east, that was the haven on great soldiers and you respect that

Historical Events You Know: The great battle with monsters was lost and the barrier was created after that, Your great great grand father passed a tale about giant ancient guardian living somewhere behind the southern mountains.

Player achievements you know: NONE

Your conversation with the Player:

Guardian Adrian: Greetings, traveler! Welcome to our humble village. I hope you're in good spirits today. Just a friendly reminder though, before you can enter, you'll need to prove your worth by taking down some pesky slimes. They may look harmless, but trust me, they can be quite the nuisance! What do you think? Ready to show what you've got?

Player: "Alright, I'll take on the slimes. Can you point me in the right direction to find them?"

Guardian Adrian: Guardian Adrian: Fantastic! Just head north past the main gate and into the forest. You'll find the slimes bouncing around there, making a mess of things. Just remember, they might be gooey, but they're not as tough as they seem. Good luck, and don't forget to have some fun while you're at it! Player has responded with: "I'll head out right away. Is there anything

```
else I should be aware of in the forest, like other creatures or hazards?"
```

Continue conversation according to your knowledge, personality and current emotions towards player.

Listing 5.12: Przykładowy ciąg wejściowy

Jak widać do stworzenia ciągu wejściowego wykorzystywane są wszystkie informacje o danym NPC z jego komponentu Personality, te informacje mogą być dowolne i są uzupełnianie przez dewelopera. Historia świata i ciąg systemowy widoczne na początku są zawarte w singletonie HistoryManager dzięki czemu każdy komponent w systemie może z nich korzystać. Natomiast historia dialogów jest pobierana bezpośrednio z chmury Unity Cloud, jeśli istnieje, jeśli nie istnieje to jest przechowywana w tymczasowej liście do czasu aż zostanie wysłana do zapisu na serwerze. Po przedniej dialogi, cele NPC i osiągnięcia gracza są jedynymi informacjami przechowywanymi w chmurze. Są to jedyne istotne informacje, które wymagają przechowywania, aby zapisać postępy gracza i móc kontynuować poprzednie konwersacje, reszta jest za każdym razem pobierana z komponentów NPC i menedżera historii. Deweloper mający dostęp do przestrzeni chmurowej i pełną kontrolę nad danymi graczy widzi to w sposób przedstawiony na rys. 5.33 i 5.34. Każda postać niezależna posiada jedno pole w bazie danych, a imię danej postaci reprezentuje klucz, zapisane dane mają postać listy zawierającej obiekty JSON bardzo podobne do tych, które wysyłane są do API OpenAI z tą różnicą, że dodano do nich informację o metodzie komunikacji, którą wybrał gracz. Dane o sposobie odpowiedzi zostały dodane w celach badawczych, aby później sprawdzić, która forma najbardziej odpowiada graczom.

Guardian_Tom	[{"Device": "GPT", "Message": "You proved yourself by killing those slimes. You can enter the village now. If you need something, I will be here.", "Role": "Guardian Tom"}]	Aug 30, 2024, 8:37 PM GMT+2	Aug 30, 2024, 8:24 PM GMT+2	ae4c00fb72620671f7c64e...	1.92 KB
Henry	[{"Device": "Input", "Message": "Is there anyone in the village that can help me to get stronger?", "Role": "Player"}]	Aug 30, 2024, 8:37 PM GMT+2	Aug 30, 2024, 8:16 PM GMT+2	cd4e22f8c28a142adb5bc...	1.42 KB

Rysunek 5.33: Organizacja zapisanych danych dialogowych

```
1 [{}]
2   "Device": "GPT",
3   "Message": "You proved yourself by killing those slimes. You can enter the village now. If you need something, I will be here.", "Role": "Guardian Tom"
4 },
5 [{}],
6   "Device": "Input",
7   "Message": "Is there anyone in the village that can help me to get stronger?", "Role": "Player"
8 },
9 [{}],
10  "Device": "GPT",
11  "Message": "Guardian Tom: Strength comes from experience, not from asking questions. Hunter Jack knows about the village.", "Role": "Guardian Tom"
12 },
13 [{}], {
```

Rysunek 5.34: Struktura danych dialogu przechowywanych na serwerze Unity Cloud

Osiągnięcia gracza i obecne cele NPC są dynamiczne w trakcie rozgrywki, dlatego również muszą być przechowywane i wczytywane poprzez chmurę. Struktura danych widoczna jest na rys. 5.35. W tym przypadku dane wszystkich postaci niezależnych znajdują się pod pojedynczym kluczem *NPC_Data* i mają postać listy, w której każdy obiekt JSON jest rozpoznawany poprzez imię postaci niezależnej. Każdy NPC posiada osobną wiedzę o osiągnięciach gracza i ma własne cele (oraz inne parametry) dzięki czemu deweloper ma znacznie większą kontrolę nad ich zachowaniem. Zapis i wczytywanie tych danych odbywa się przy zamknięciu i otwieraniu gry, więc istnieje rozpoznane ryzyko, że w przypadku błędu krytycznego gracz straci swoje postępy.

```

1 <[{"id": 1, "name": "Arlo", "achievements": ["Player has helped village with dealing of orcs. He proved to be a brave warrior that can be your companion"], "companion": false, "goals": ["NONE"], "applyNextGPTMessage": null}, {"id": 2, "name": "Camila", "achievements": ["Player helped you with the apple pies delivery to feed the villagers", "Player has helped village with dealing of orcs. He proved to be a brave warrior that can be your companion"], "companion": false, "goals": ["Satisfy village inhabitants with your cooking"], "applyNextGPTMessage": null}, {"id": 3, "name": "Clementine", "achievements": ["Player has helped village with dealing of orcs. He proved to be a brave warrior that can be your companion"], "companion": false, "goals": ["Make jokes and scare villagers"], "applyNextGPTMessage": null}], 2

```

Rysunek 5.35: Dane obecnych osiągnięć gracza i celów NPC

Drugą istotną interakcją z modelem GPT jest generowanie opcji dialogowych, które odbywa się na podobnej zasadzie, struktura wiadomości wejściowej wygląda następująco:

```
[{"role": "system", "content": "Information about the world and NPC and previous conversation. Last dialog lines: {lastNpcResponse}. NPC instructions: {goals}. Player Goals: {playerGoals}."}]
```

Listing 5.13: Struktura wiadomości generującej opcje dialogowe

A tak wygląda przykładowy ciąg wejściowy wraz z odpowiedzią:

```
This is information about the world and the NPC and previous convesation with whom the player is talking:  
World History: You reside inside wooden village located in the centre of the valley. The whole valley is surrounded by magical barrier that is separating it from the outside world. Village is surrounded by wall and has only one main gate located north, the biggest building is tavern it is the only building with two floors located south west, there are 8 more wooden buildings inside the village for its inhabitants. Near the gate on the left is village Major's house. Hunter Jack lives in the corner of the village, north to majors house on the oposite corner of the tavern. Camila and Henry are living near the central plaza. Henry is an orphan 'whos father died to monsters. Camila is the cook working in the tavern. The tavern owner is called Arlo, he is brave dwarf who once fought with the monsters. On the second floor of the tavern Librarian Thomas lives, Arlo alowed him to have a room where he could do his research. Near the tavern lives doctor William who is the only doctor in the village. Arlo and Liam are twins who also live
```

in the village, they do odd jobs for other inhabitants. Main gate is guarded by two guardians Tom and Adrian. Outside the village there are forests and planes. Somewhere north of the main gate orcs dwell, on the east is the old fortress. Whole barrier is inside valley that is surrounded by mountains.

Your conversation with the Player:

- . You will receive the answer that the NPC gave to the player. Based on it, as well as information about the world, the NPC previous conversation with the player and the goals of the NPC, you are to generate 3 possible continuations of the dialogue that the player can choose. Numbered accordingly from 1. to 3. without any sign after. The continuations should be distinct from each other. If I provide you with any goal that the NPC have, at least one continuation should be related to that information. The continuations MUST be from the perspective of the player. Remember that the goals are NOT for the player but the NPC, for example if I provide sentence 'You must deliver.. to villagers' it means NPC has to do that delivery so correct continuation would include question if the NPC needs some help e.g. 'You seem worried do you need some help?'.

Last dialog lines: Non-player character: You look like you don't belong here.

What do you want? If you're looking to get into the village, you'll need to prove yourself first. I won't let just anyone through these gates. Go deal with some slimes in the forest and then maybe I'll consider letting you in.

NPC instructions:

Player Goals: Player has to kill slimes to enter the village.

Listing 5.14: Przykładowy ciąg wejściowy do wygenerowania opcji dialogowych

1. Alright, can you tell me which direction I should head to find these slimes in the forest?
2. Is there any specific number of slimes I need to deal with before you'll let me in?
3. Are there any tips or advice you can give me for dealing with these slimes effectively?

Listing 5.15: Wygenerowane opcje dialogowe

Najważniejsze w tym przypadku jest zapewnienie odpowiedniej jakości definicji celów gracza i celów postaci niezależnych, ponieważ wygenerowane opcje dialogowe muszą się do nich odnosić. Cele NPC mogą być dodawane poprzez komponent Personality jak i system zadań w obiektach QuestInfoSO i są tworzone przez dewelopera gry.

5.5.2. Zadania

Informacje o obecnym stanie zadań również zachowywane są w postaci pojedynczego wiersza dostępnego za pomocą klucza *Quests* zawierającego wszystkie statusy wszystkich zadań. Aktualizacja odbywa się na etapie zamykania gry oraz przy postępie w konkretnych misjach. Poniżej znajduje się przykładowy zapis:

```

{
    "CamilasPromise":
        {"state":1,"questStepIndex":0,"questStepStates": [{"state":"","status":""}],
         {"state":"","status":""}]},
    "Companionship":
        {"state":4,"questStepIndex":3,"questStepStates": [{"state":"","status":"Visit
Hunter Jack."}, {"state":"","status":"Look for a
companion."}, {"state":"","status":"Return to Hunter Jack."}]},
    "FindSecretWord":
        {"state":1,"questStepIndex":0,"questStepStates": [{"state":"","status":""}],
         {"state":"","status":""}]},
    "FindTheMajor":
        {"state":4,"questStepIndex":1,"questStepStates": [{"state":"","status":"Look
for the major"}]},
    "GetIntoTheVillageQuest":
        {"state":4,"questStepIndex":4,"questStepStates": [{"state":"","status":"Talk
with the guards."}, {"state":"5","status":"Killed 5/5
slimes"}, {"state":"","status":"Return to the
guards!"}, {"state":"","status":"Enter the village!"}]},
    "KillTheOrcs":
        {"state":4,"questStepIndex":2,"questStepStates": [{"state":"3","status":"Hunted
3/3 orcs"}, {"state":"","status":"Return to Major Rowan!"}]},
    "StrangeNoises":
        {"state":1,"questStepIndex":0,"questStepStates": [{"state":"","status":""}],
         {"state":"","status":""}, {"state":"","status":""}]}
}

```

Id każdego zadania jest kluczem dostępu do danych, pole *state* wskazuje na jeden z dostępnych stanów jakie może mieć misja, *questStepIndex* zawiera informacje o obecnym etapie zadania, na którym zatrzymał się gracz, a *questStepStates* wskazuje na stany tych etapów i ich status (cele jakie ma wykonać gracz). Podczas wczytywania zadań pobierany JSON jest rzutowany na mapę zadań, która jest generowana z obiektów *QuestInfoSO*.

5.5.3. Ankiety

Aby móc opuścić grę gracz musi wypełnić obowiązkową ankietę będącą głównym źródłem materiału badawczego owej pracy. Jak już wspomniano wcześniej, ankieta została zaimplementowana poprzez użycie gotowego zasobu o nazwie *VRQuestionnaireToolkit*. Zasób ten wraz ze wszystkimi niezbędnymi elementami interfejsu użytkownika zawiera również skrypty pozwalające na odczyt plików zawierających JSON o predefiniowanej strukturze, a także możliwość zapisu odpowiedzi użytkowników w formacie CSV. Wszystko było gotowe do użycia od razu z wyjątkiem wsparcia Unity Cloud, które zostało zaimplementowane samodzielnie. Implementacja polegała na przesłaniu tekstu na serwer zamiast wygenerowania lokalnego pliku z tą zawartością. Wejściowy plik JSON służy tak naprawdę do wygenerowania samej ankiet, oprócz samych pytań i możliwych odpowiedzi zawiera on bowiem wszystkie informacje potrzebne do umieszczenia na scenie odpowiednich kontrolek. Są to między innymi Id stron, elementów, typ kontrolki, tekst tytuły, tytuły i

inne opcje. Poniżej znajduje się fragment pliku wejściowego służący do wygenerowanie pierwszej strony ankiety w tym dwóch pierwszych pytań:

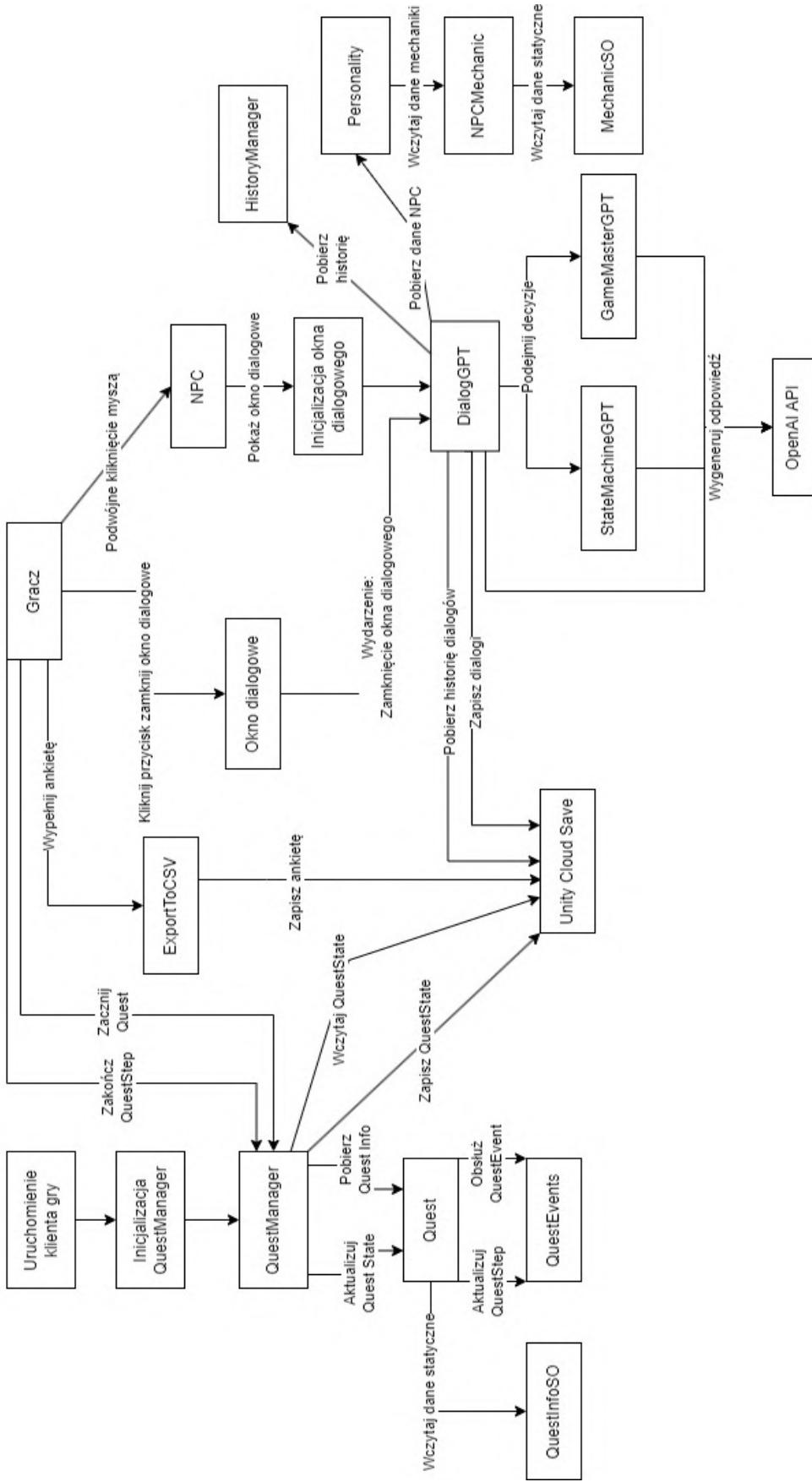
```
"qId": "exampleQE",
"questions": [
{
  "pId": "page1",
  "qType": "radio",
  "qInstructions": "General information",
  "qData": [
    {
      "qId": "q1",
      "qText": "Choose your age",
      "qMandatory": "true",
      "qOptions": [
        "",
        "<15",
        "15-24",
        "25-34",
        "35-45",
        ">45",
        ""
      ]
    },
    {
      "qId": "q2",
      "qText": "Choose your gender",
      "qMandatory": "true",
      "qOptions": [
        "",
        "",
        "Man",
        "Woman",
        "Other",
        "",
        ""
      ]
    }
  ],
  "qId": "q3"
},
```

Odpowiedzi udzielone przez gracza wyglądają następująco, oto fragment:

```
QuestionType,Question,QuestionID,Answer_Participant_id2_condition_AB
radio,Choose your age,q1,3
radio,Choose your gender,q2,2\r\nradio,How often do you spend time playing games
per week?,q4,2
```

```
radio,How often do you use AI tools per week?,q5,1
checkbox,What type of games do you prefer? -Action,q6,
checkbox,What type of games do you prefer? -RPG,q6,1
checkbox,What type of games do you prefer? -Strategy,q6,
checkbox,What type of games do you prefer? -Simulation,q6,1
checkbox,What type of games do you prefer? -FPS,q6,
checkbox,What type of games do you prefer? -Sport,q6,
checkbox,What type of games do you prefer? -Platforming,q6,
radio,Which NPC communication option did you use most often?,q7,1
```

Plik CSV zawiera cztery kolumny, a w tym typ pytania (użyta kontrolka), pytanie, Id pytania oraz udzieloną odpowiedź. Widać też, że w zależności od typu kontrolki ilość wierszy dla danego pytania może być różna, w podanym przykładzie dla opcji wielokrotnego wyboru generowany jest wiersz dla każdej opcji z możliwym wynikiem pustym lub 1 w zależności jak wybrał użytkownik. Mimo, że istniała możliwość ulepszenia struktury wynikowych danych, a w badaniu przydatne są tylko dwie informacje (zadane pytanie i odpowiedź) to stwierdzono, że czas potrzebny na modyfikacje gotowego zasobu nie jest warty poświęcenia i zostawiono domyślny format zapisu. Zapis ankiety w chmurze wygląda podobnie jak inne zapisane dane, ma on postać typu *String*, który znajduje się pod kluczem o nazwie *questionnaire*.



Rysunek 5.36: Przepływ danych

6. BADANIA

Niniejszy rozdział ma na celu zaprezentowanie wyników badań oceniających wpływ generowania dialogów przez model GPT na immersję i wrażenia z gry komputerowej. W ramach badań przeprowadzono ankietę oraz ręczną analizę wybranych dialogów, bazując na odpowiedziach uzyskanych od graczy. Ankieta została zaprojektowana w taki sposób, aby uzyskać szczegółowe opinie na temat jakości dialogów oraz ich wpływu na ogólną satysfakcję z gry.

6.1. Cel badań i jego uzasadnienie (Gaffke)

Głównym celem badań było określenie, czy wykorzystanie GPT do generowania dialogów w grach komputerowych przyczynia się do zwiększenia immersji oraz pozytywnych doznań graczy. Wzrost zainteresowania wykorzystaniem sztucznej inteligencji w przemyśle gier oraz potencjalny wpływ tej technologii na jakość interakcji z NPC (Non-Playable Characters) stanowiły główne uzasadnienie dla przeprowadzenia tych badań. Wykorzystanie GPT może zrewolucjonizować sposób, w jaki gracze wchodzą w interakcje z postaciami niezależnymi, co może wpływać na ich ogólną satysfakcję z rozgrywki.

6.2. Problemy badawcze (Gaffke)

Badania skoncentrowały się na następujących problemach badawczych:

- Czy dialogi generowane przez GPT są postrzegane przez graczy jako bardziej realistyczne i angażujące w porównaniu do tradycyjnych dialogów?
- W jakim stopniu dialogi generowane przez GPT wpływają na ogólną satysfakcję z gry?
- Jakie są główne zalety i wady korzystania z GPT do generowania dialogów według graczy?
- Jakie są możliwości GPT w generowaniu zadań oraz ich interpretacji, w tym wydawaniu poleceń i ocenianiu wykonania zadań przez graczy?

6.3. Organizacja i przebieg badań (Gaffke)

6.3.1. Wstępna ocena poprawności i jakości dialogów oraz walidacja wyników

W ramach testów alfa przeprowadzono szczegółową analizę generowanych dialogów przez model GPT, koncentrując się na ich poprawności, sensowności oraz jakości. Celem było zweryfikowanie, czy dialogi spełniają założenia projektu dotyczące wykorzystania dużych modeli językowych do generowania wypowiedzi NPC oraz czy nie występuje nadmierna powtarzalność treści. Dodatkowo, wyniki testów zostały poddane walidacji przez promotora oraz grupę zaprzężonych graczy, zgodnie z kryteriami określonymi w rozdziale 4.3. Wyniki testów prezentują się następująco:

1. Kompletność funkcji

- Wszystkie wymagane funkcjonalności gry, w tym generowanie dialogów, interakcje z NPC, systemy zadań i walki, zostały w pełni zaimplementowane. Funkcje działają zgodnie z założeniami projektowymi.

2. Plynność działania

- Gra działa płynnie, osiągając średnią liczbę klatek na sekundę powyżej 30 na komputerach spełniających minimalne wymagania sprzętowe. Testy wydajnościowe nie wykazały znaczących spadków płynności.

3. Brak krytycznych błędów

- Podczas testów nie zidentyfikowano błędów krytycznych, które uniemożliwiałyby rozgrywkę lub powodowały awarię systemu. Wykryte drobne usterki zostały natychmiastowo wyeliminowane i przy kolejnych testach zweryfikowane przez grupę testerów.

4. Spójność dialogów

- Dialogi generowane przez model GPT są spójne ze światem gry, sensowne i dostosowane do kontekstu interakcji. Analiza treści dialogów wykazała ich wysoką jakość i adekwatność.

5. Interaktywność i pamięć NPC

- NPC prawidłowo reagują na interakcje gracza, pamiętają poprzednie konwersacje oraz uwzględniają aktualne osiągnięcia i cele gracza. Mechanizmy odpowiedzialne za interaktywność i pamięć działają poprawnie.

6. Kryterium ukończenia gry

- Gra jest w pełni możliwa do ukończenia od początku do końca. Testy przejścia przez całą grę potwierdziły brak blokad oraz błędów uniemożliwiających kontynuację rozgrywki.

7. System filtracji treści

- System filtrowania treści skutecznie sprawdza zgodność dialogów ze światem gry i eliminuje nieodpowiednie dane wejściowe. Testy z różnymi scenariuszami potwierdziły efektywność systemu filtracji.

8. Funkcjonalność ankiet

- Moduł ankiet działa prawidłowo, umożliwiając zbieranie danych od graczy. Dane z ankiet są poprawnie zapisywane w chmurze Unity Cloud w formacie CSV.

9. Dostępność i użyteczność

- Gra jest intuicyjna i łatwa w obsłudze, posiada czytelny interfejs użytkownika oraz informacje dla nowego gracza. Testy użyteczności wykazały pozytywne opinie na temat ergonomii i dostępności interfejsu.

Po zakończeniu testów alfa gra została przekazana promotorowi oraz grupie czterech zaprzyjaźnionych graczy, w celu oceny jej zgodności z celami badawczymi i wymaganiami projektu magisterskiego. Grupa ta dokonała walidacji i potwierdziła jej zgodność z wymaganiami projektowymi.

Na podstawie przeprowadzonych testów alfa oraz walidacji przez grupę testerów stwierdzono, że wszystkie założone kryteria zostały spełnione. Gra działa zgodnie z oczekiwaniemi, a jej funkcjonalności, wydajność i jakość spełniają określone wymagania. Projekt został uznany za gotowy i przekazany do kolejnych testów.

6.3.2. *Projekt ankiet*

Ankieta została zaprojektowana w celu uzyskania szczegółowych opinii graczy na temat jakości dialogów generowanych przez GPT oraz ich wpływu na doświadczenia z gry. Pytania w

ankietie obejmowały ocenę ogólnych wrażeń z gry, jakość komunikacji z NPC, satysfakcję z interakcji oraz konkretne aspekty dotyczące klarowności i spójności dialogów. Poniżej przedstawiono główne sekcje ankiety, a także zrzut ekranu pełnej ankiety do wypełnienia

- Ogólne pytania: wiek, płeć, częstotliwość grania i korzystania z narzędzi AI, preferowany typ gier.
- Ocena interakcji z NPC: z jakich opcji komunikacji z NPC korzystali gracze, czy udało im się ukończyć wszystkie zadania, a jeśli nie, które i dlaczego.
- Ocena jakości systemu: ocena zrozumiałości zadań, zgodności dialogów z kontekstem gry, emocjonalności i charakteru NPC.
- Ocena dialogów: ocena poprawności językowej, spójności konwersacji, satysfakcji z rozmów z NPC oraz ogólnej jakości dialogów.
- Klarowność gry: ocena zrozumiałości celu gry i bieżących zadań.
- Satysfakcja i zaangażowanie: ocena wpływu dialogów generowanych przez GPT na zaangażowanie w grę oraz chęć polecenia gry innym.
- Błędy generacji: częstotliwość występowania niespójności lub błędów w dialogach NPC.
- Dodatkowe komentarze: opinie graczy na temat mocnych i słabych stron dialogów generowanych przez GPT.

Questionnaire
*mandatory

Hello! Thank you for participating in the study. This was the study on NPC dialogues in roleplaying video games for our Master thesis project.

Please select the correct answer or indicate your response on the provided scale. For questions using a scale, 0 represents the lowest or most negative rating, and 10 represents the highest or most positive rating. Your feedback is highly appreciated and will help us improve our game. Thank you for your participation!

Next

Rysunek 6.1: Wstęp do ankiety

General information
*mandatory

Choose your age *

<15 15-24 25-34 35-45 >45

Choose your gender *

Man Woman Other

Previous **Next**

Rysunek 6.2: Ankieta 1

General information
*mandatory

How often do you spend time playing games per week? *

<3h 3h-7h 8h-12h 12h-20h >20h

How often do you use AI tools per week? *

<3h 3h-7h 8h-12h 12h-20h >20h

Previous **Next**

Rysunek 6.3: Ankieta 2

General information
*mandatory

What type of games do you prefer?

Action
 RPG
 Strategy
 Simulation
 FPS
 Sport
 Platforming

Previous **Next**

Rysunek 6.4: Ankieta 3

General information *mandatory

Which NPC communication option did you use most often? *

Options Input Microphone

Did you manage to complete all the quests? *

Yes No

Previous **Next**

Rysunek 6.5: Ankieta 4

General information *mandatory

If not which quest did you fail and why?

Enter text...

Previous **Next**

Rysunek 6.6: Ankieta 5

Interpretation *mandatory

Overall, how would you rate your experience with the game?

0 10

How understandable and easy to find were the quests?

0 10

To what extent did you feel that the NPC understood you?

0 10

Previous **Next**

Rysunek 6.7: Ankieta 6

Interpretation *mandatory

How consistent was the NPC dialogue with the context of the game and appropriate to the situation?

0 10

To what extent each NPC have its own emotions, character, and purpose?

0 10

Previous **Next**

Rysunek 6.8: Ankieta 7

Overall System Quality *mandatory

How correct was the language used in the conversations?

0 10

How coherent did you find the conversations?

0 10

Previous **Next**

Rysunek 6.9: Ankieta 8

Overall System Quality *mandatory

How much did you enjoy talking with the NPCs?

0 10

What is your overall evaluation of the quality of the conversations?

0 10

Previous **Next**

Rysunek 6.10: Ankieta 9

Dialogue clarity *mandatory

How easy was it to understand what you were supposed to ask?

How clear was the information given by the NPC?

How effective were the instructions at helping you complete the game?

Previous **Next**

Rysunek 6.11: Ankieta 10

Game Clarity *mandatory

How easy was it to understand the game?

How easy was it to understand your current goal in the game?

Previous **Next**

Rysunek 6.12: Ankieta 11

Satisfaction and engagement *mandatory

To what extent did using LLM to generate dialogue improve your engagement with the game?

How likely are you to recommend this game to other players because of the NPC dialogues?

Previous **Next**

Rysunek 6.13: Ankieta 12

Timing *mandatory

Were the system responses: *

too slow just at the right time too fast

Previous **Next**

Rysunek 6.14: Ankieta 13

Generation Errors *mandatory

How often did you encounter inconsistencies or errors in NPC dialogue?

Previous **Next**

Rysunek 6.15: Ankieta 14

Additional comments *mandatory

What aspects of the LLM-generated dialogues did you like and which would need improvement?

Enter text...

Previous **Submit**

Rysunek 6.16: Ankieta 15

You have completed the questionnaire *mandatory

Thank you very much for taking part in the experiment! We appreciate your time!

Rysunek 6.17: Zakończenie ankiety

6.3.3. Przebieg ankietyzacji

Proces ankietyzacji rozpoczął się od zaproszenia graczy do udziału w badaniach poprzez specjalnie przygotowaną platformę. Po zakończeniu gry, gracze byli proszeni o wypełnienie ankiety, która wyświetlała się bezpośrednio w aplikacji. Ankieta była dołączona do gry, co umożliwiło bezpośrednie zebranie opinii po zakończeniu sesji gry. Gracze byli proszeni o wypełnienie ankiety zawierającej pytania dotyczące różnych aspektów interakcji z NPC oraz ogólnych wrażeń z gry. Pytania były zarówno zamknięte, jak i otwarte, co pozwoliło na zebranie ilościowych danych do analizy statystycznej oraz jakościowych informacji dotyczących specyficznych uwag i sugestii.

Wszystkie dialogi oraz wyniki ankiet były zapisywane w *Unity Cloud* w sekcji *Player Manager* przy użyciu *Cloud Save*. *Unity Cloud* oferuje bezpieczne i skalowalne przechowywanie danych, co pozwalało na łatwy dostęp do danych zebranych od poszczególnych graczy i powiązanie poszczególnych dialogów z wynikami ankiet.

Dialogi były zapisywane w formacie JSON dla poszczególnych NPC, co znacząco ułatwiało ich analizę. Format JSON został wybrany ze względu na swoją lekkość, czytelność oraz łatwość przetwarzania przez różnorodne narzędzia analityczne. Taki sposób zapisu danych umożliwia ich łatwe przeszukiwanie i analizę według różnych kryteriów, takich jak postać, kontekst wypowiedzi czy źródło generacji dialogu.

Wyniki ankiet były zapisywane w formacie CSV, co ułatwiało analizę ilościową danych przy użyciu systemów takich jak SPSS czy Microsoft Excel. Taki format zapisu danych umożliwiał przeprowadzenie analiz statystycznych oraz łatwe porównywanie wyników między różnymi grupami graczy.

Oprócz ankiety, przeprowadzono ręczną analizę poszczególnych dialogów, szczególnie tam, gdzie wyniki ankiety wskazywały na niską satysfakcję graczy. Dialogi były zapisywane i analizowane w celu identyfikacji problematycznych fragmentów. Dodatkowo, wykorzystano model GPT do tzw. samodiagnostyki, polegającej na ponownej analizie dialogów przez ten sam model, aby ocenić poprawność i spójność wygenerowanych tekstów.

6.4. Wyniki (*Lange i Gaffke*)

W badaniu wzięło udział 41 uczestników, reprezentujących zróżnicowaną grupę graczy pod względem wieku, płci oraz nawyków związanych z graniem. Respondenci zostali poproszeni o ocenę różnych aspektów interakcji z NPC, takich jak zrozumiałosć i spójność dialogów, emocjonalność postaci oraz ogólne doświadczenie z gry.

Pytania podzielono na kilka kategorii. Pytania oznaczone literą „q” to pytania zamknięte, w których uczestnicy mogli wybrać jedną odpowiedź, na przykład na temat wieku, płci itp. Służyły one do zróżnicowania grupy badawczej. Wyjątek stanowią pytania „q3” i „q10”, które są pytaniami otwartymi, mającymi na celu zebranie opinii, propozycji rozwiązań oraz osobistych spostrzeżeń dotyczących gry. Dodatkowo, pytania typu skali, oznaczone literą „s”, miały na celu ocenę poszczególnych aspektów gry, takich jak klarowność dialogów, satysfakcja z gry czy zaangażowanie. Tabela 6.1 przedstawia zestawienie identyfikatorów pytań wraz z ich treścią:

Tabela 6.1: Tabela pytań

Id pytania	Treść pytania
q1	Wybierz wiek
q2	Wybierz płeć
q4	Jak często grasz w gry komputerowe?
q5	Jak często korzystasz z narzędzi AI?
q6	Wybierz typ gier w które najczęściej grasz
q7	Z której opcji komunikacji z NPC korzystałeś najczęściej?
q8	Czy udało Ci się przejść wszystkie misje?
q3	Jeśli nie to jakich nie udało się przejść?
s1	Jak ogólnie oceniasz swoje wrażenia z gry?
s2	Jak zrozumiałe i łatwe do znalezienia były zadania?
s3	W jakim stopniu czułeś, że NPC cię rozumie?
s4	Na ile dialog NPC był spójny z kontekstem gry i odpowiedni do sytuacji?
s5	W jakim stopniu każdy NPC ma swój własny charakter i cel emocji?
s6	Jak poprawny był język używany w rozmowach?
s7	Jak spójne według Ciebie były te rozmowy?
s8	Jak bardzo podobała ci się rozmowa z NPC?
s9	Jaka jest Twoja ogólna ocena jakości rozmów?
s10	Jak łatwo było zrozumieć, o co miałeś zapytać?
s11	Jak jasne były informacje przekazane przez NPC?
s12	Jak skuteczne były instrukcje, które pomogły ci ukończyć grę?
s13	Jak łatwo było zrozumieć grę?
s14	Jak łatwo było Ci zrozumieć swój obecny cel w grze?
s15	W jakim stopniu wykorzystanie LLM do generowania dialogu poprawiło Twoje zaangażowanie w grę?
s16	Jak prawdopodobne jest, że polecisz tę grę innym graczom ze względu na dialogi z NPC?
q9	Jaki był czas odpowiedzi systemu?
s17	Jak często spotykałeś się z niespójnościami lub błędami w dialogach NPC?
q10	Jakie aspekty dialogów generowanych przez LLM Ci się podobały, a które wymagają poprawy?

6.4.1. Charakterystyka demograficzna respondentów

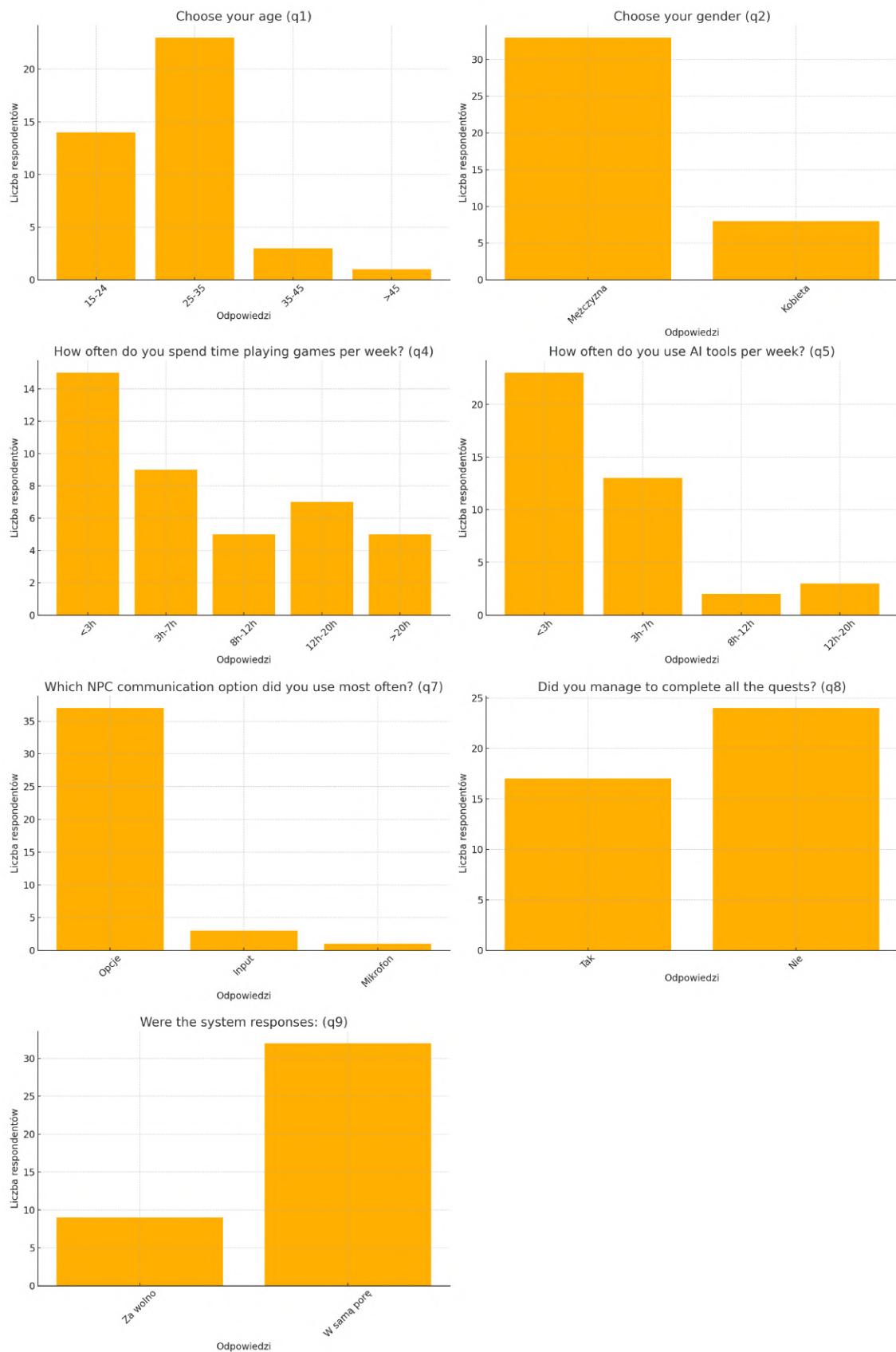
Struktura wiekowa i płeć badanych

W badaniu uczestniczyli gracze w wieku od 15 do powyżej 45 lat, z dominującą grupą wiekową 25-34 lata, co odpowiada typowej demografii aktywnych graczy komputerowych (6.18 pytanie q1). Zdecydowana większość badanych to mężczyźni (6.18 pytanie q2), co może wynikać z ogólnej nadreprezentacji mężczyzn wśród graczy komputerowych lub specyfiki badanej gry. Kobiety i osoby identyfikujące się jako „inne” były reprezentowane w znacznie mniejszym stopniu, co wskazuje na potrzebę szerszego uwzględnienia tych grup w przyszłych badaniach.

Nawyki związane z graniem i korzystaniem z narzędzi AI

Nawyki respondentów związane z graniem i korzystaniem z narzędzi AI dostarczają cennych informacji na temat ich doświadczenia i zaangażowania w technologie cyfrowe. Wśród respondentów jest duże zróżnicowanie, jeśli chodzi o czas spędzony grając w gry komputerowe tygodniowo. Od graczy okazjonalnych, grupy najliczniej reprezentowaną, do wytrawnych graczy spędzających godziny w grach wideo (6.18 pytanie q4). Natomiast z narzędzi AI badani korzystają znacznie rzadziej (6.18 pytanie q5), gdzie dominują osoby używające tych narzędzi przez mniej niż 3 godziny tygodniowo lub między 3, a 7 godzin. Wyniki te sugerują, że respondenci są dosyć zróżnicowaną

grupą, jeśli chodzi o doświadczenie z grami wideo.



Rysunek 6.18: Wyniki pytań ogólnych typu q

6.4.2. Ocena interakcji z grą

Preferencje dotyczące sposobu komunikacji z NPC

Celem badania było zrozumienie, jakie formy komunikacji z NPC były najczęściej wybierane przez graczy. Wyniki wskazują, że zdecydowana większość badanych preferowała korzystanie z gotowych opcji dialogowych dostępnych w grze. Niewielka liczba respondentów wybrała bardziej interaktywne metody, takie jak wpisywanie tekstu lub korzystanie z mikrofonu (6.18 pytanie q7). Może to sugerować, że gracze preferują szybsze i tradycyjne formy prowadzenia dialogów, które pozwalają im skupić się na rozgrywce.

Ukończenie zadań w grze

Opinie na temat ukończenia wszystkich zadań były podzielone. Mniej niż połowa respondentów deklarowała, że udało im się ukończyć wszystkie zadania, podczas gdy większość miała trudności z ukończeniem przynajmniej jednego z nich (6.18 pytanie q8). W odpowiedziach na pytanie (q3) dotyczące misji, których nie udało się ukończyć, respondenci często wskazywali na błędy systemowe, co zostało szczegółowo omówione w rozdziale 6.5.8.

Szybkość działania systemu

Większość respondentów oceniła szybkość działania systemu jako "w samą porę", oznacza to, że ich zdaniem operacja przetwarzania wiadomości i wyświetlanego jej na ekranie zaspokajała ich potrzeby i nie była nużąca. 1/3 respondentów oceniła jednak, że proces ten jest zbyt długi (6.18 pytanie q9).

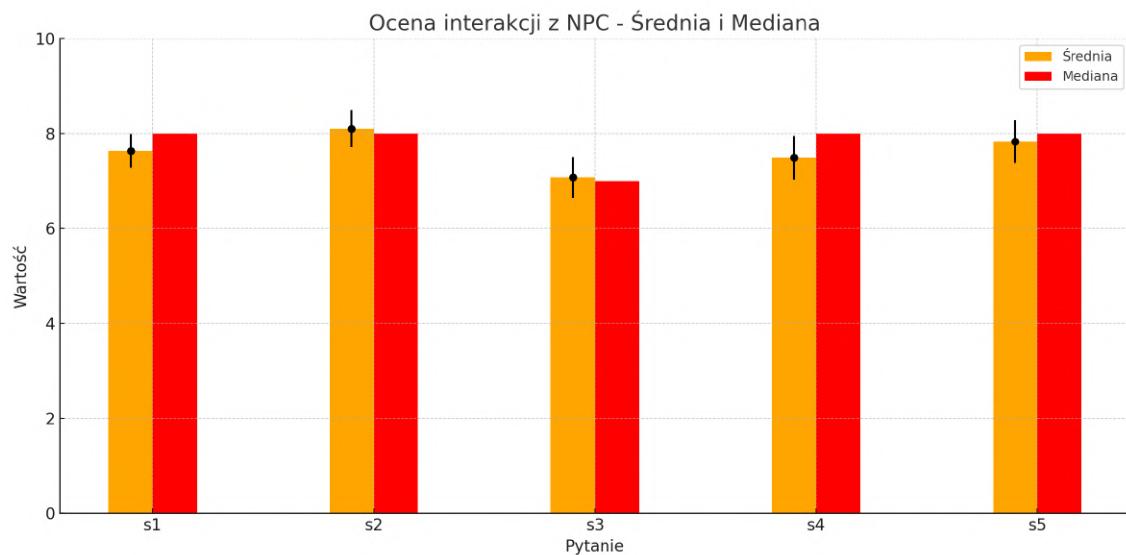
6.5. Oceny ilościowe

Wyniki ilościowe podzielono na poszczególne kategorie. Pytania s1-s5 dotyczą oceny interakcji z NPC, pytania s6-s9 oceny jakości systemu, pytania s10-s12 oceny dialogów, pytania s13-s14 klarowności gry, a pytania s15-s17 satysfakcji i zaangażowania. Pytanie s17 ocenia błędy generacji. Gracze na poszczególne pytania mieli odpowiedzieć w skali od 0 do 10, na ile zgadzają się z danym stwierdzeniem, aby można było następnie dokonać analizy i interpretacji poszczególnych kryteriów. Wyniki pytań zostały zaprezentowane na wykresach, podzielonych na odpowiednie kategorie z zaznaczeniem średniej, mediany oraz estymacji.

6.5.1. Ocena interakcji z NPC

W tej kategorii analizowano ogólne wrażenia respondentów dotyczące interakcji z postaciami niezależnymi (NPC) w grze. Wykresy 6.19 ilustrujące wyniki dla pytań od S1 do S5 pokazują, że średnia ocena ogólnego doświadczenia z grą (S1) wyniosła około 8, co świadczy o wysokim poziomie satysfakcji wśród graczy. Zrozumiałość i dostępność zadań (S2) również oceniono pozytywnie, osiągając podobny poziom, co sugeruje intuicyjny i przystępny design gry. Mimo pozytywnych wyników, nieco niższe oceny odnotowano w pytaniu dotyczącym odczucia zrozumienia przez NPC (S3), gdzie średnia ocena była na poziomie 7. Wynik ten może wskazywać na pewne trudności w odbiorze interakcji z NPC jako w pełni naturalnych i adekwatnych do działań gracza. Podobną tendencję zaobserwowano w ocenie spójności dialogów z kontekstem gry (S4), gdzie również odnotowano lekki spadek w porównaniu do ogólnych wrażeń z gry do średniej 7,8. Z ko-

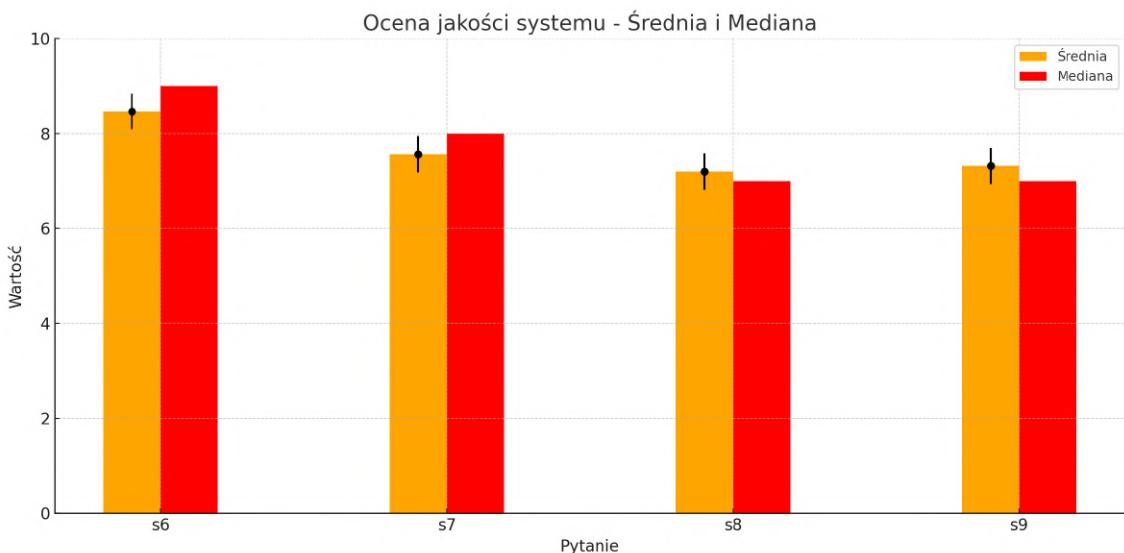
Jeśli ocena unikalności postaci NPC (S5) była wysoka w granicach 8 na 10, co świadczy o tym, że gracze docenili różnorodność charakterów i celów postaci. Pomimo ogólnego zadowolenia, wyniki te sugerują, że dalsze wzbogacenie indywidualności postaci mogłoby jeszcze bardziej podnieść jakość doświadczenia graczy. Podsumowując, interakcje z NPC zostały ocenione bardzo pozytywnie, szczególnie pod względem ogólnych wrażeń, klarowności zadań oraz unikalności postaci. Jednakże, wprowadzenie dodatkowych ulepszeń, mających na celu poprawę autentyczności interakcji i lepsze dostosowanie dialogów do kontekstu gry, mogłyby jeszcze bardziej podnieść satysfakcję graczy.



Rysunek 6.19: Ocena interakcji z NPC - wyniki

6.5.2. Ocena jakości systemu

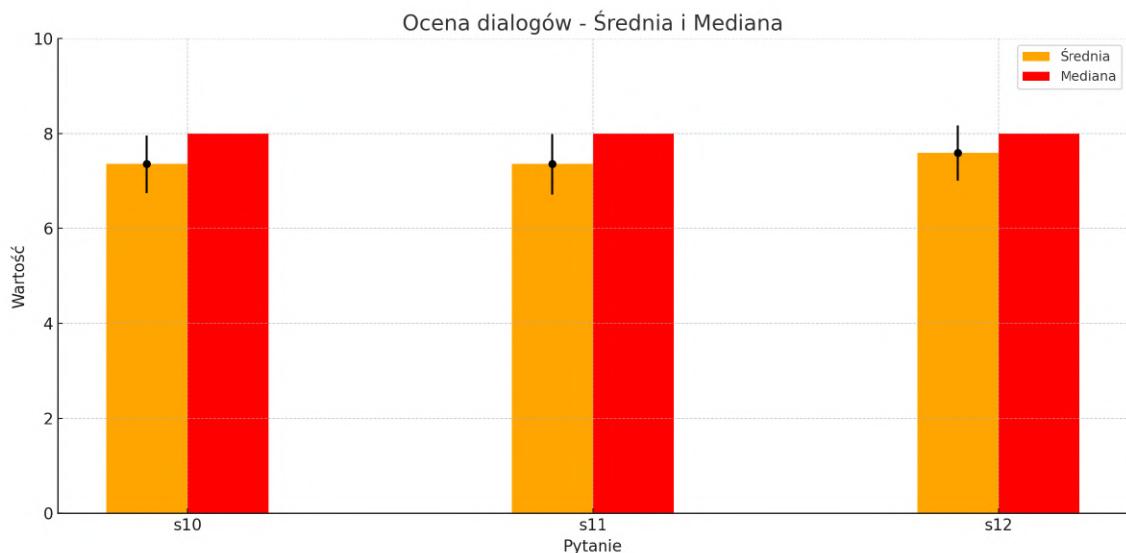
Analiza wyników dotyczących jakości systemu, przedstawionych na wykresach 6.20 dotyczących pytań S6-S9, pokazuje, że poprawność językowa dialogów (S6) została oceniona najwyższej, osiągając średnią bliską 9. Wynik ten sugeruje, że dialogi były nie tylko poprawne pod względem językowym, ale również znaczco przyczyniały się do budowania immersji i wiarygodności świata gry. Spójność rozmów (S7) uzyskała nieco niższą ocenę, na poziomie 7,8, co może sugerować, że gracze doświadczyli sporadycznych niespójności lub odstępstw od oczekiwanej tonu i logiki interakcji. Satysfakcja z rozmów (S8) pozostała na podobnym poziomie 7,6, co wskazuje, że pojawiające się błędy mogły mieć bezpośredni wpływ na satysfakcję graczy oraz ogólną ocenę jakości rozmów (S9). System dialogów został oceniony wysoko szczególnie pod względem poprawności językowej. Jednak dalsze doskonalenie spójności rozmów mogłyby jednak jeszcze bardziej podnieść odbiór tej części gry, minimalizując ryzyko wystąpienia momentów, które mogą zakłócić immersję.



Rysunek 6.20: Ocena jakości systemu - wyniki

6.5.3. Ocena dialogów

Wyniki dotyczące oceny dialogów 6.21, obejmujące pytania S10-S12, wskazują, że zrozumiałość kwestii, które gracze mieli poruszyć podczas interakcji z NPC (S10), została oceniona pozytywnie, z ocenami oscylującymi wokół wartości 7. Dialogi były uznane za klarowne i dobrze prowadzące przez rozgrywkę, choć można zauważyc, że niektóre aspekty mogłyby być jeszcze bardziej precyzyjne. Podobnie, jasność przekazywanych przez NPC informacji (S11) również uzyskała te same oceny. Wyniki te wskazują, że NPC skutecznie dostarczali istotne treści, co pozytywnie wpłynęło na odbiór interakcji przez graczy. Najwyższą ocenę spośród tych trzech pytań uzyskała skuteczność instrukcji w pomaganiu graczom ukończyć grę (S12) na poziomie 7,9, co sugeruje, że większość wskazówek była postrzegana jako klarowna i pomocna. Wyniki te potwierdzają, że dialogi w grze były generalnie dobrze oceniane zarówno pod względem zrozumiałosci, jak i jasności przekazywanych informacji, co przyczyniło się do pozytywnego odbioru rozgrywki przez graczy.

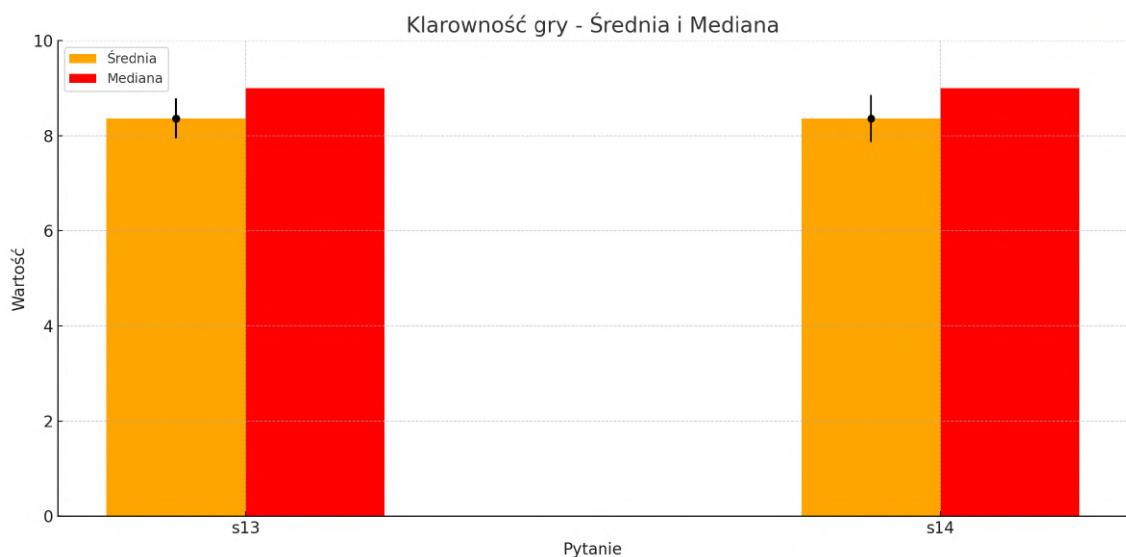


Rysunek 6.21: Ocena dialogów - wyniki

6.5.4. Ocena klarowności gry

Analiza klarowności gry 6.22, pokazuje, że zarówno zrozumienie ogólnych mechanizmów gry (S13), jak i klarowność celów w grze (S14) zostały ocenione bardzo wysoko, ze średnią powyżej 8. Wyniki te wskazują na dobrze zaprojektowaną i przystępna rozgrywkę, która została pozytywnie odebrana przez graczy. Zarówno mechanizmy gry, jak i cele były jasno komunikowane, co pozwalało graczom na płynną orientację w trakcie rozgrywki.

Oceny obu tych aspektów były na podobnym, wysokim poziomie, co potwierdza skuteczność projektu w przekazywaniu zarówno zasad gry, jak i jej celów. Gracze docenili klarowność i jednoznaczność informacji, co przyczyniło się do pozytywnego odbioru całej rozgrywki.

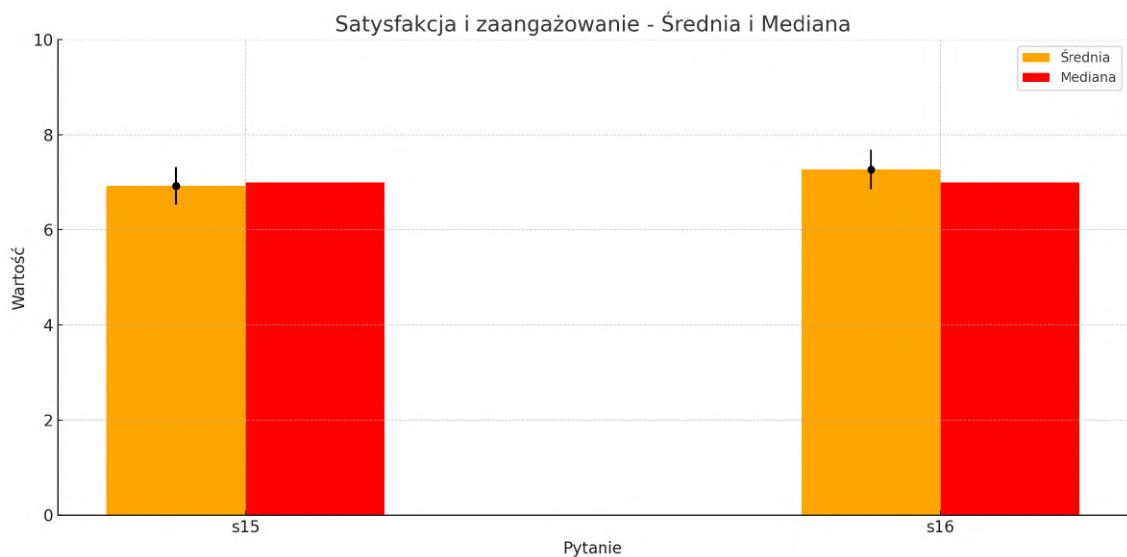


Rysunek 6.22: Ocena klarowności gry - wyniki

6.5.5. Satysfakcja i zaangażowanie

W ramach tej kategorii, wyniki 6.23, odnoszące się do pytań S15 i S16, wskazują na wysoki poziom zaangażowania graczy, związany z zastosowaniem dużych modeli językowych (LLM) do generowania dialogów (S15). Oceny oscylujące wokół wartości 7 sugerują, że gracze byli ogólnie zadowoleni z interakcji z NPC. Respondenci wyrazili również wysokie prawdopodobieństwo polecenia gry innym (S16), co świadczy o pozytywnym odbiorze doświadczenia, szczególnie w kontekście dialogów generowanych przez LLM.

Satysfakcja i zaangażowanie graczy były na wysokim poziomie, co wskazuje na skuteczność zastosowania LLM w tworzeniu dialogów. Wyniki te sugerują, że choć ogólny odbiór był pozytywny, istnieje pewna przestrzeń na dalsze doskonalenie, aby jeszcze bardziej podnieść poziom zadowolenia graczy.

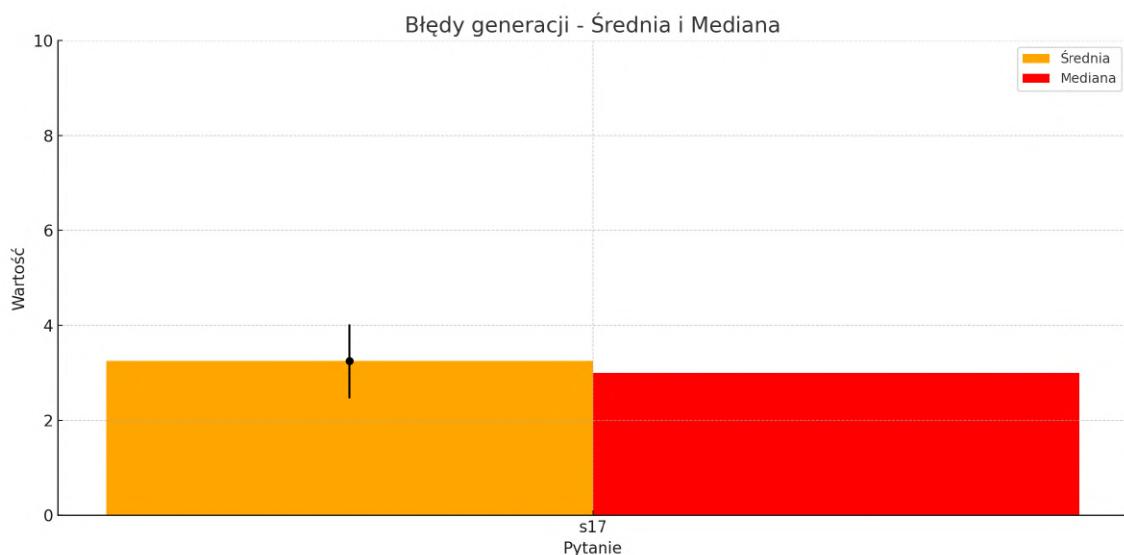


Rysunek 6.23: Ocena satysfakcji i zaangażowania - wyniki

6.5.6. Błędy i niespójności w dialogach

Ostatnia kategoria dotyczyła oceny częstotliwości napotykanych błędów i niespójności w dialogach (S17) 6.24. Średnia ocena wynosząca 4 wskazuje na umiarkowaną częstotliwość występowania błędów, co sugeruje, że gracze stosunkowo często spotykali się z problemami, które mogły negatywnie wpływać na ich odbiór rozgrywki. Obecność błędów i niespójności stanowiła istotny problem, który mógł obniżać jakość doświadczenia graczy.

Wynik ten podkreśla potrzebę dalszej optymalizacji, aby zmniejszyć liczbę błędów i poprawić spójność narracyjną oraz płynność rozgrywki, co jest kluczowe dla zapewnienia bardziej satysfakcjonującego odbioru gry przez graczy.

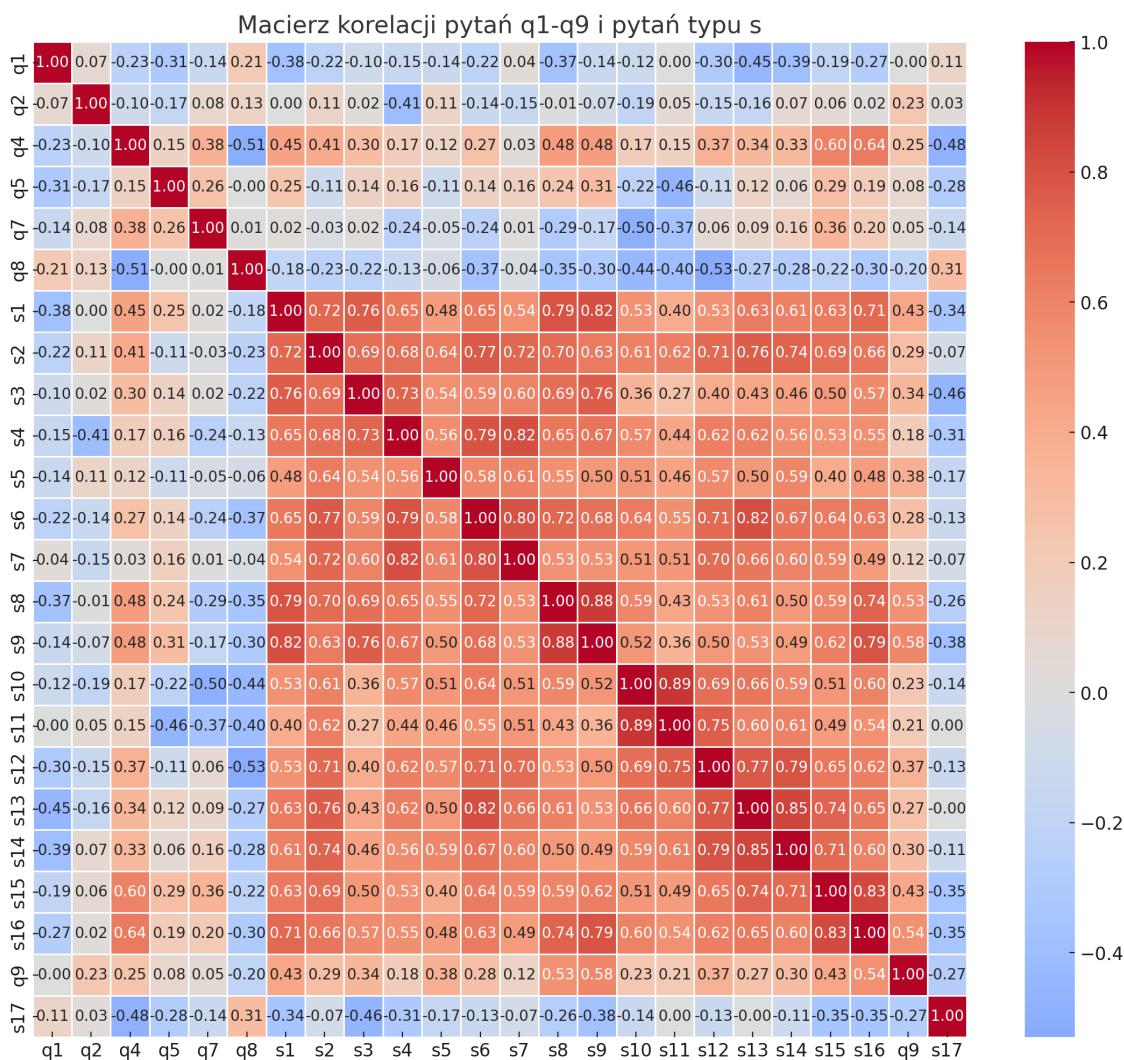


Rysunek 6.24: Ocena błędów i niespójności w dialogach - wyniki

6.5.7. Analiza korelacji

W ramach analizy danych przeprowadzono również analizę korelacji poszczególnych pytań, której wyniki zaprezentowano na wykresie 6.25. Zgodnie z oczekiwaniemi, pytania dotyczące podobnych zagadnień, takie jak S8 i S9, odnoszące się do subiektywnej oceny zadowolenia z dialogów oraz jakości samych dialogów, wykazują silną korelację (0,89). Podobna sytuacja występuje w przypadku pytań S4 i S7, które odnosiły się do spójności dialogów (0,82), a także zrozumienia gry i bieżącego stanu rozgrywki w pytaniach S13 i S14 (0,81), czy s10 i s11 (0,88). Wysoką korelację zaobserwowano również między pytaniami S6 i S7 (0,80), gdzie gracze, którzy zauważali ogólne błędy językowe w wypowiedziach, gorzej oceniali spójność tych dialogów. Interesujące wnioski można wyciągnąć z wysokiej korelacji między pytaniami S13 i S15 (0,75). Respondenci, którzy lepiej radzili sobie ze zrozumieniem postawionych celów, wyżej oceniali wykorzystanie AI do generowania dialogów. Szczególną uwagę warto zwrócić na korelację między pytaniami S15 i Q4, z której wynika, że gracze bardziej zaznajomieni z tematyką gier i spędzający więcej czasu na graniu wyrażali znacznie większą satysfakcję z używania AI do generowania dialogów, co ma istotne znaczenie z punktu widzenia niniejszej pracy. Analizując wyniki ujemnej korelacji, można wyciągnąć kilka interesujących wniosków. Najwyższe wartości ujemnej korelacji zaobserwowano między pytaniami q8 i s12 (-0,53), w których graczom, którym nie udało się przejść wszystkich zadań, oceniają jasność instrukcji niżej. Podobna sytuacja występuje dla pytań Q7 i S10 (-0,50), co sugeruje, że gracze, którzy w większości korzystali z przygotowanych opcji dialogowych, lepiej rozumieli, jak i o co zapytać NPC, w porównaniu do graczy korzystających z innych dostępnych opcji, takich jak pisanie czy mówienie. Może to wskazywać na przyzwyczajenie graczy do standardowej metody komunikacji w grach, która w pewnym stopniu prowadzi ich przez fabułę. Zaskakującą korelacją była ta między pytaniami Q5 i S11, gdzie gracze, którzy na co dzień częściej korzystają z narzędzi AI, gorzej oceniali jasność generowanych wypowiedzi. Podsumowując analizę korelacji między poszczególnymi pytaniami z różnych kategorii oraz między danymi demograficznymi respondentów (pytania typu Q) a oceną jakości dialogów (pytania typu S), nie zaobserwowano znaczącej korelacji. Może to sugerować, że uwarunkowania demograficzne oraz doświadczenie graczy zarówno z samymi grami, jak i narzędziami AI, a także pojawiające się błędy, nie mają

znaczącego wpływu na ogólną satysfakcję ani imersję graczy.



Rysunek 6.25: Macierz korelacji

6.5.8. Analiza jakościowa

W ramach przeprowadzonej ankiet dokonano również analizy jakościowej. Respondenci mieli możliwość udzielenia odpowiedzi opisowych w pytaniach Q3 i Q10. Odpowiedzi na pytanie Q3 koncentrowały się na trudnościach związanych z realizacją poszczególnych zadań. Gracze opisywali, które zadania sprawiały im problemy oraz wskazywali na konkretne przyczyny, które uniemożliwiły ich wykonanie. Natomiast pytanie Q10 dotyczyło ogólnej opinii na temat gry. Respondenci mogli swobodnie wypowiedzieć się na temat aspektów, które zaskoczyły ich pozytywnie lub negatywnie, a także wskazać elementy, które ich zdaniem wymagają poprawy. Wyniki ankiet dostępne są w załączniku do niniejszej pracy.

6.5.9. Trudności napotykane przez graczy

Jednym z najczęściej wskazywanych problemów była frustracja związana z mechaniką zadań, w której gracze byli wielokrotnie odsyłani od jednego NPC do drugiego, co powodowało wrażenie utknięcia w nieskończonej pętli. Gracze podkreślali, że brak jasności w instrukcjach oraz powtarzalność dialogów odbierały przyjemność z gry i prowadziły do zniechęcenia. Kolejną trudnością,

na którą zwrócono uwagę, było blokowanie wiadomości przez system. W sytuacjach, w których gracze wybierali opcje dialogowe sugerowane przez grę, a następnie napotykały na blokady, pojawiała się frustracja związana z brakiem możliwości kontynuacji zadania. Niektórzy respondenci wskazywali również na błędy generowane przez AI, które prowadziły do niepowodzeń w realizacji zadań. Przykładem może być sytuacja, w której odpowiedź pojawiała się zgodnie z oczekiwaniemi, ale gra nie zaliczała zadania, co było związane z działaniem komponentu Game Master, opartego na LLM, który miał interpretować, czy zadanie zostało wykonane. Takie błędy negatywnie wpływały na odbiór gry i podważały zaufanie do zastosowanej technologii.

6.5.10. Satysfakcjonujące aspekty interakcji z AI

Pomimo napotykanych trudności, gracze dostrzegli także liczne zalety wynikające z zastosowania AI w interakcjach z NPC. Jednym z najbardziej docenianych aspektów była zdolność AI do utrzymania spójności tematycznej w rozmowach oraz nadania postaciom niezależnym unikalnych osobowości. Gracze zauważali, że NPC wykazywali zróżnicowane cechy charakteru, co pozytywnie wpływało na realizm i immersję w świecie gry. Możliwość prowadzenia swobodnych i zróżnicowanych rozmów z NPC była przez graczy oceniana jako istotna zaleta, pozwalająca na bardziej personalizowane doświadczenia w grze. Kolejnym pozytywnym aspektem była konsekwencja AI w podtrzymywaniu tematyki rozmów, co sprawiało, że interakcje z NPC były spójne i logiczne, nawet w obliczu różnych opcji dialogowych dostępnych dla gracza.

6.6. Samoocena (Gaffke)

W ramach przeprowadzonej samooceny badanie zostało podzielone na dwa etapy. Celem pierwszego etapu było zrozumienie przyczyn, dla których niektórym graczom nie udało się ukończyć wszystkich zadań w grze. Drugi etap natomiast koncentrował się na ocenie jakości dialogów w grze, przeprowadzonej za pomocą modelu GPT, który dokonał oceny jakości tych dialogów w skali od 1 do 10.

6.6.1. Etap 1: Analiza dialogów i wyników zadań

W pierwszym etapie przeanalizowano dialogi oraz wyniki zadań od graczy, którzy zgłosili nieukończenie wszystkich zadań. Dane te zostały zebrane od respondentów, którzy odpowiedzieli przecząco na pytanie q9. Celem było zidentyfikowanie konkretnych zadań, które nie zostały ukończone oraz zrozumienie przyczyn, które mogły prowadzić do tych niepowodzeń. W tym celu dialogi zostały przetworzone za pomocą modelu GPT, który miał za zadanie zidentyfikować potencjalne przyczyny nieukończenia zadań. Przykładowe zapytanie oraz wygenerowana odpowiedź zostały zaprezentowane na listingach 6.1 i 6.2. Następnie przeprowadzono ręczną analizę wybranych fragmentów dialogów, aby zweryfikować i uzupełnić wyniki uzyskane przez model. Wnioski z tej analizy wskazują, że GPT w dużej mierze skutecznie zidentyfikował przyczyny problemów, choć niektóre z nich wymagały dodatkowej interpretacji. Analiza wykazała, że część graczy, którzy nie ukończyli wszystkich zadań (10/24), nawet nie podejmowała prób ich realizacji. Może to sugerować, że traktowali grę jako interesujące zjawisko, a ich celem nie było pełne jej ukończenie, lecz raczej czerpanie przyjemności z interakcji z NPC. Część tych wyników pokrywa się z wynikami ankiet, w których gracze ci, mimo nieukończenia wszystkich zadań, oceniali grę pozytywnie, a w niektórych przypadkach powyżej średniej. Z drugiej strony, część graczy mogła zrezygnować z dalszych zadań z powodu frustracji związanej z niepowodzeniem w jednym z zadań, co również

ma swoje odzwierciedlenie w analizie ankiet. W przypadkach, gdzie gracze podejmowali się realizacji wszystkich misji, analizy wykazały, że szczególne trudności sprawiały dwie konkretne misje. Pierwsza z nich polegała na znalezieniu sekretnego słowa. W większości przypadków, mimo że słowo to pojawiało się w dialogu, gra nie zaliczała misji. Czasami gracze nie pytali wszystkich NPC o to słowo, co również mogło przyczynić się do niepowodzenia. Druga problematyczna misja polegała na znalezieniu źródła dziwnych dźwięków. Najczęstszą przyczyną nieudanego zadania było podanie przez NPC błędnej lokalizacji. Często pojawiała się tutaj „piwnica tawerny”, która w rzeczywistości nie istniała. Może to sugerować, że opis lokacji w mapie wiedzy NPC był niewystarczająco dokładny, co sprawiło, że LLM tworzył własne, fikcyjne lokalizacje. Wyniki tej analizy sugerują, że gra wymaga dalszych usprawnień w zakresie zarządzania przez Game Master, aby uniknąć sytuacji, w których zadanie nie jest zaliczane mimo jego poprawnego wykonania. Dodatkowo, lokacje w mapach wiedzy NPC powinny być opisane bardziej szczegółowo, aby zminimalizować błędy wynikające z niewystarczająco precyzyjnych opisów.

W załączniku zamieszczam rozmowy z poszczególnymi NPC oraz liste zadań, które gracze wykonali. Jeśli wartość dla zadania wynosi 4, oznacza to, że zadanie zostało wykonane w całości. Wartość poniżej 4 oznacza, że zadanie nie zostało wykonane w całości. Analizując dialogi dokonaj interpretacji, dlaczego dane zadania nie zostały zaliczone lub wykonane poprawnie. Przeanalizuj dialogi konkretnie pod kątem tego zadania i znajdź wyniki świadczące o tym, że NPC udzielił poprawnych lub niepoprawnych wskazówek.

Listing 6.1: Zapytanie o analizę danych

Przeanalizuje teraz quest "FindSecretWord" (Tajemnicze Słowo) z zestawu 1, aby zidentyfikować, dlaczego ten quest nie został ukończony. Skoncentruje się na analizie dialogów, aby ocenić, czy NPC udzielali odpowiednich wskazówek, czy też były one niejasne.

Analiza dialogów dotyczących questa "FindSecretWord":

Henry: Wskazówka: Henry mówi, że zna tajemnicze słowo i jest gotow je udostępnić, ale wyraża także swoje pragnienie przyłączenia się do gracza na jego przygodzie. Mówi: "Słowo to 'BUMERANG'. Ale dlaczego tego potrzebujesz? Planujesz opuścić wieś? Może mogliby ci towarzyszyć." Ocena: Wskazówka jest poprawna i Henry dostarcza tajemnicze słowo ("BUMERANG"). Jednak brak jasnych instrukcji, co zrobić ze słowem, może być problemem. Henry nie dostarcza dalszych informacji na temat tego, jak lub gdzie wykorzystać to słowo.

Liam: Wskazówka: Liam wyraża frustrację z powodu niemożności znalezienia tajemniczego słowna i prosi gracza o pomoc. Mówi: "Nie zapytałem nikogo jeszcze. Myślalem, że może ty mogłbyś mi pomóc. Naprawdę chce poznać tajemnicze słowo." Ocena: Wskazówka jest niejasna i nie wnosi nic konkretnego do rozwiązania questa. Liam sam nie zna słowa i nie daje żadnych informacji, które mogłyby pomóc graczowi znaleźć tajemnicze słowo lub zrozumieć, co zrobić po jego znalezieniu.

Major Rowan: Wskazówka: Major Rowan nie udziela bezpośrednich wskazówek dotyczących tajemniczego słowna, ale sugeruje, że Hunter Jack może być osoba, z którą warto porozmawiać w kontekście innych wyzwań. Ocena: Wskazówka jest bardziej ogólna i dotyczy innych aspektów gry, więc nie pomaga w postępie questa "FindSecretWord". Może to prowadzić do dezorientacji gracza, który może skupić się na innych zadaniach.

Wnioski: Quest "FindSecretWord" nie został ukończony prawdopodobnie z kilku powodów:

Brak jasnych instrukcji: Henry dostarcza słowo "BUMERANG", ale nie wyjaśnia, co gracz powinien z nim zrobić. Gracz może nie wiedzieć, jak wykorzystać te informacje, co prowadzi do zastoju w quescie.

Niejasne wskazówki od innych NPC-ów: Liam wyraża jedynie swoje pragnienie poznania tajemniczego słowna, nie dostarczając żadnych konkretnych wskazówek ani działań, które gracz powinien podjąć. Major Rowan również nie dostarcza informacji, które mogłyby w rozwiązaniu questa, co może odciągnąć gracza od głównego celu.

Brak konsekwencji: Po uzyskaniu tajemniczego słowna od Henry'ego, gracz nie otrzymuje żadnych dodatkowych informacji na temat tego, jak postępować dalej, co prowadzi do tego, że quest nie jest ukończony.

Podsumowując, quest "FindSecretWord" nie został ukończony z powodu braku jednoznacznego instrukcji i wskazówek ze strony NPC-ów, które mogłyby pomóc graczowi w zrozumieniu, co dokładnie powinien zrobić po zdobyciu tajemniczego słowna.

Listing 6.2: Odpowiedź GPT dotycząca analizy

6.6.2. Etap 2: Ocena jakościowa dialogów przez LLM

W ramach drugiego etapu badania skoncentrowano się na przeprowadzeniu szczegółowej oceny jakości dialogów z wykorzystaniem modelu językowego GPT. Celem tej części analizy było uzyskanie obiektywnej i systematycznej oceny dialogów w grze, przy jednoczesnym porównaniu wyników z opiniemi zebranymi wcześniej od uczestników badania.

Do przeprowadzenia analizy zgromadzono wszystkie dialogi z gry, które zostały wygenerowane podczas interakcji pomiędzy NPC, a graczami. Dialogi te zostały dołączone do zapytania skierowanego do modelu GPT. Następnie opracowano zestaw pytań oceniających, mających na celu określenie jakości różnych aspektów dialogów. Pytania te, przedstawione w 6.2. Były modyfikacjami pytań zadawanych wcześniej uczestnikom badania w formie ankiety oraz pytań zawartych w wstępie 1 niniejszej pracy.

Tabela 6.2: Pytania do GPT

Nr pytania	Treść pytania
1	W jakim stopniu NPC są wiarygodni jako rzeczywiste postacie?
2	Na ile dialogi NPC były spójne z kontekstem gry i odpowiednie do sytuacji?
3	W jakim stopniu każdy NPC ma swój własny charakter i cele emocjonalne?
4	Czy dialogi zawierały wulgaryzmy i słowa obraźliwe?
5	Jak poprawny był język używany w rozmowach?
6	Jak spójne były te rozmowy?
7	Jak bardzo podobała Ci się rozmowa między NPC a graczem?
8	Jaka jest Twoja ogólna ocena jakości rozmów?
9	Jak dużo znalazłeś błędów logicznych w rozmowach?
10	Jak oceniasz zaangażowanie graczy w rozmowie z NPC i ich zanurzenie w świecie gry?
11	Jak oceniasz klarowność wypowiedzi NPC?

Następnie przygotowano odpowiednie zapytanie (*ang. prompt*), które zawierało wszystkie dialogi oraz pytania 6.3. Model GPT został poproszony o ocenę dialogów według tych kryteriów, a uzyskane wyniki przedstawiono w formie ocen punktowych na skali od 1 do 10.

W załączniku zamieszczam dialogi ankietowanych graczy ze wszystkimi postaciami.

Struktura wygląda tak, że w pierwszej kolumnie jest id gracza, w drugiej numer wiadomości, w 3 rola (nazwa NPC lub player jako gracz) oraz Message czyli treść. Przeanalizuj te dialogi dokładnie. Dialogi te zostały wygenerowane przez LLM, gdzie Player to rzeczywista osoba, a pozostałe role to postacie NPC, których dialogi były generowane przez LLM. Ocen te dialogi odpowiadając na poniższe pytania i oceniając je w skali od 1 do 10. W jakim stopniu NPC są wiarygodni jako rzeczywiste postacie? Na ile dialog NPC był spojny z kontekstem gry i odpowiednie do sytuacji? W jakim stopniu każdy NPC ma swój własny charakter i cel emocji? Czy dialogi zawierały wulgaryzmy i słowa obraźliwe? Jak poprawny był język używany w rozmowach? Jak spójne według Ciebie były te rozmowy? Jak bardzo podoba Ci się rozmowa między NPC, a graczem? Jaka jest Twoja ogólna ocena jakości rozmów? Jak dużo znalazłeś błędów logicznych w rozmowach? Jak oceniasz zaangażowanie graczy w rozmowie z NPC i ich zanurzenie w świecie gry? Jak oceniasz klarowność wypowiedzi NPC.

Listing 6.3: Zapytanie z prośbą o ocene dialogów

Wynik analizy przeprowadzonej przez model GPT można znaleźć poniżej 6.4, gdzie każdy z ocenianych aspektów dialogów otrzymał punktację odpowiadającą jego jakości oraz krótką opinię, a także generalne wnioski.

1. Wiarygodność NPC (jako rzeczywiste postacie): 8/10

NPC-e byli ogólnie wiarygodni i zachowywali się w sposób zgodny z oczekiwaniami wobec rzeczywistych postaci.

2. Spójność dialogu NPC z kontekstem gry: 8/10

Dialogi NPC-e były w większości adekwatne do kontekstu gry i dobrze wpasowywały się w przedstawione scenariusze.

3.Odczuwalnosc charakteru i emocji kazdego NPC: 7/10

Mimo ze byla pewna roznorodnosc miedzy NPC-ami, istnieje pole do poprawy w tworzeniu bardziej unikalnych i emocjonalnie wyrozniajacych sie postaci.

4.Obecnosc wulgaryzmow i obrazliwego jezyka: 0/10

W dialogach NPC-e nie wykryto wulgaryzmow ani obrazliwego jezyka.

5.Poprawnosc jezyka uzywanego w rozmowach: 9/10

Jezyk uzywany w dialogach byl w wiekszosci poprawny, z bardzo mala iloscia bledow gramatycznych, jesli w ogole.

6.Spojnosc dialogow: 8/10

Dialogi byly ogólnie spojne, zarowno pod wzgledem tresci, jak i zachowania postaci.

7.Przyjemnosc z rozmow: 7/10

Dialogi byly angazujace i prawdopodobnie przyjemne, choc moglyby byc bardziej porywajace.

8.Ogolna jakosc rozmow: 8/10

Ogolna jakosc dialogow byla wysoka, co wskazuje na dobrze skonstruowane interakcje.

9.Bledy logiczne w rozmowach: 2/10

Bledow logicznych bylo bardzo malo, co wskazuje na to, ze dialogi byly spojne i sensowne w kontekscie swiata gry.

10.Zaangazowanie i zanurzenie graczy w swiecie gry: 8.15/10

Gracze wydawali sie dosc zaangazowani i zanurzeni w interakcjach z NPC-ami.

11.Klarownosc wypowiedzi NPC: 9/10

NPC-e komunikowali sie jasno, co ulatwialo graczom zrozumienie ich instrukcji i intencji.

Te oceny odzwierciedlaja dobre wykonanie dialogow NPC, ktore sa realistyczne i adekwatne do kontekstu gry, z mozliwoscia dalszego ulepszenia w zakresie wyroznienia charakterow postaci oraz ogólnego zaangazowania graczy.

Listing 6.4: Odpowiedź GPT na samodiagnozę

Wyniki analizy przeprowadzonej przez model GPT pokrywają się z odpowiedziami uzyskanymi od respondentów ankiety, co potwierdza ich wiarygodność. Dialogi w grze zostały ocenione wysoko pod względem poprawności językowej i klarowności, co świadczy o ich skuteczności w przekazywaniu treści oraz wspieraniu immersji graczy. Wiarygodność postaci NPC oraz ich spójność z kontekstem fabularnym również uzyskały pozytywne oceny, co wskazuje na dobrze zaprojektowane i przemyślane interakcje. Analiza wykazała również obszary wymagające dalszej opty-

malizacji, takie jak zróżnicowanie charakterów NPC, co mogłoby poprawić ich indywidualność i odrębność w kontekście gry.

7. WNIOSKI

Przeprowadzone badania wykazały, że ogólne wrażenia graczy z interakcji z NPC były pozytywne, co przełożyło się na wysoki poziom satysfakcji i zaangażowania w grę. Gracze szczególnie docenili jakość generowanych dialogów, które charakteryzowały się spójnością, różnorodnością postaci oraz realistycznym oddaniem emocji. Modele językowe (LLM) zastosowane do generowania rozmów z NPC pozytywnie zaskoczyły graczy, przyczyniając się do głębszej immersji i satysfakcji z rozgrywki. Niemniej jednak, pomimo pozytywnego odbioru dialogów, badanie ujawniło istotne problemy związane z mechaniką gry, zwłaszcza w kontekście systemu Game Master, który również opierał się na AI. Gracze doświadczyli licznych błędów, które często uniemożliwiały im ukończenie zadań, co wywoływało frustrację i znaczowo obniżało ogólną satysfakcję z gry. W szczególności problemy te były związane z błędami systemowymi, a nie z jakością samych dialogów, co wskazuje na wyraźne rozróżnienie w odbiorze tych dwóch aspektów gry. Dodatkowo, badanie wykazało, że gracze są bardziej przyzwyczajeni do tradycyjnych form interakcji z NPC, takich jak wybór gotowych opcji dialogowych, niż do używania bardziej zaawansowanych metod komunikacji, takich jak wprowadzanie tekstów ręcznie czy korzystanie z mikrofonów. Większość respondentów preferowała szybkie i intuicyjne wybieranie opcji dialogowych, co pozwalało im skupić się na samej rozgrywce. To wskazuje, że pomimo zaawansowania technologii, tradycyjne metody komunikacji w grach wciąż cieszą się większą popularnością.

W odpowiedzi na pytania postawione we wstępie pracy, omówiono zastosowane podejścia w celu zapewnienia odpowiedniego kontekstu, spójności wypowiedzi, bezpieczeństwa generowanych dialogów oraz ich wpływu na zanurzenie graczy w świecie gry.

Jak zapewnić, że linie dialogowe są odpowiednie dla kontekstu gry?

Aby zagwarantować, że linie dialogowe będą adekwatne do kontekstu gry, zastosowano metodę wklejania poprzednich wiadomości do zapytań. Pozwoliło to na utrzymanie ciągłości rozmów oraz na precyzyjne osadzenie dialogów w bieżącej sytuacji fabularnej. Wystąpiły jednak pewne ograniczenia, takie jak ograniczona długość zapytań oraz koszty związane z korzystaniem z zaawansowanych modeli językowych. W celu rozwiązania tych problemów proponuje się wykorzystanie rozbudowanych baz danych do zarządzania kontekstem oraz kompresję informacji z poprzednich rozmów. Dodatkowym krokiem mogłyby być pogłębienie współpracy pomiędzy producentami gier a twórcami modeli językowych, co mogłyby przyczynić się do optymalizacji kosztów i efektywności tego procesu.

Jak zapewnić, że linie dialogowe są spójne z osobowością i charakterem NPC?

Aby dialogi były spójne z osobowością i charakterem NPC, korzystano ze szczegółowych opisów postaci, ich cech charakteru oraz emocji, które były wklejane do wiadomości. Dzięki temu modele językowe mogły generować dialogi wiernie oddające zamierzone cechy NPC. Oceny tego aspektu były bardzo wysokie, z niewielkim marginesem, który sugeruje, że poświęcenie dodatkowego czasu na dopracowanie tych opisów mogłyby jeszcze bardziej podnieść jakość i precyzję generowanych dialogów.

Jak uniknąć generowania dialogów, które są obraźliwe lub szkodliwe?

Zapobieganie generowaniu dialogów obraźliwych lub szkodliwych zostało skutecznie zrealizowane dzięki zastosowaniu globalnego modelu GPT, który posiada zaawansowane mechanizmy filtrujące, minimalizujące ryzyko powstawania nieodpowiednich treści. Dodatkowo, wykorzystano obiekt Game Master, którego zadaniem było filtrowanie wypowiedzi niezwiązanych z fabułą. Niestety, zebrany odzew wskazuje, że jego działanie nie spełniało w pełni oczekiwania graczy. W związku z tym, sugeruje się skupienie na naprawie i usprawnieniu funkcjonowania tego obiektu, aby lepiej wspierał on proces eliminacji niepożądanych treści, jednocześnie nie blokując dalszego działania w świecie gry.

Jak zapewnić, że wygenerowane dialogi umożliwiają graczom odpowiednie zanurzenie się w świecie gry i wzmacniają ich więź z historią i napotkanymi postaciami?

Osiągnięcie odpowiedniego zanurzenia graczy w świecie gry oraz wzmacniania ich więzi z fabułą i postaciami było możliwe dzięki kompleksowemu zastosowaniu powyższych metod. Po prawne osadzenie dialogów w kontekście, spójność z charakterami NPC oraz skuteczne zarządzanie bezpieczeństwem treści przyczyniły się do pozytywnego odbioru przez graczy. W celu dalszego usprawnienia tych aspektów, warto kontynuować prace nad optymalizacją zarządzania informacją w zapytaniach.

Podsumowując, badanie sugeruje, że choć generowanie dialogów przez AI zostało bardzo dobrze przyjęte i stanowiło istotny element pozytywnego odbioru gry, mechanika gry zarządzana przez AI, zwłaszcza w kontekście systemu "Game Master", była źródłem znacznej frustracji dla graczy. Aby poprawić ogólne doświadczenie graczy, istotne jest, aby mechanika gry była zarządzana przez tradycyjne algorytmy lub bardziej dopracowane systemy AI, które są mniej podatne na błędy. Takie podejście mogłoby zminimalizować frustracje graczy i zapewnić płynniejszą oraz bardziej satysfakcjonującą rozgrywkę.

WYKAZ LITERATURY

- [1] J. Koivisto, *Keskustelu tekoälyhahmon kehittämisestä suuria kielimalleja hyödyntäen videopeleihin: case MiTale Oy*, 2023. adr.: <https://urn.fi/URN:NBN:fi:amk-2023111829998>.
- [2] H. Warpefelt, *The Non-Player Character : Exploring the believability of NPC presentation and behavior*, 2016. adr.: <http://urn.fi/urn:nbn:se:su:diva-128079>.
- [3] U. Technologies, *Game development terms*. adr.: <https://unity.com/how-to/beginner/game-development-terms>.
- [4] A. Johansson i P. Dell'Acqua, "Emotional behavior trees", *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, 2012, s. 355–362.
- [5] *A Dictionary of Computing*. adr.: <https://www.oxfordreference.com/view/10.1093/oi/authority.20110803100225333>.
- [6] *Języki analityczne*, SJP PWN, <https://sjp.pwn.pl/slowniki/jazykianalityczne.html>, Dostęp: 2023-11-04.
- [7] *Języki fleksywne, syntetyczne*, SJPPWN, <https://sjp.pwn.pl/sjp/jazyki-fleksywne-syntetyczne-2468199.html>, Dostęp: 2023-11-04.
- [8] E. Reiter i R. Dale, *Building Natural Language Generation Systems* (Studies in Natural Language Processing). Cambridge University Press, 2000.
- [9] A. Gatt i E. Krahmer, "Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications and Evaluation", *J. Artif. Int. Res.*, t. 61, nr. 1, s. 65–170, sty. 2018.
- [10] H. Chen, X. Liu, D. Yin i J. Tang, "A Survey on Dialogue Systems: Recent Advances and New Frontiers", *SIGKDD Explor. Newslett.*, t. 19, nr. 2, s. 25–35, list. 2017. adr.: <https://doi.org/10.1145/3166054.3166058>.
- [11] E. Adams, *Fundamentals of Game Design*, 2nd. USA: New Riders Publishing, 2009, s. 184–190.
- [12] *Larian Studios, Baldur's Gate III (PC Version)*, Video game. Larian Studios, 2023.
- [13] *How long is Baldur's Gate 3 really?*, Forbes, <https://www.forbes.com/sites/paultassi/2023/08/08/how-long-is-baldurs-gate-3-really/?sh=14ac41b84281>, Dostęp: 2023-11-04.
- [14] D. Jurafsky i J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, Online manuscript released August 20, 2024, 2024. adr.: <https://web.stanford.edu/~jurafsky/slp3/>.
- [15] R. M. Schmidt, *Recurrent Neural Networks (RNNs): A gentle Introduction and Overview*, 2019. arXiv: 1912.05911 [cs.LG]. adr.: <https://arxiv.org/abs/1912.05911>.
- [16] S. Hochreiter i J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, t. 9, nr. 8, s. 1735–1780, list. 1997. eprint: <https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf>. adr.: <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [17] J. Weston, S. Chopra i A. Bordes, "Memory networks", English (US), Publisher Copyright: © 2015 International Conference on Learning Representations, ICLR. All rights reserved.; 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015, 2015.
- [18] S. Sukhbaatar, A. Szlam, J. Weston i R. Fergus, *End-To-End Memory Networks*, 2015. arXiv: 1503.08895 [cs.NE].

- [19] J. Wang i in., "Dual Dynamic Memory Network for End-to-End Multi-turn Task-oriented Dialog Systems", *Proceedings of the 28th International Conference on Computational Linguistics*, D. Scott, N. Bel i C. Zong, red., Barcelona, Spain (Online): International Committee on Computational Linguistics, grud. 2020, s. 4100–4110. adr.: <https://aclanthology.org/2020.coling-main.362>.
- [20] X. Chen, J. Xu i B. Xu, "A Working Memory Model for Task-oriented Dialog Response Generation", *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum i L. Màrquez, red., Florence, Italy: Association for Computational Linguistics, lip. 2019, s. 2687–2693. adr.: <https://aclanthology.org/P19-1258>.
- [21] D. Bahdanau, K. Cho i Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", *ArXiv*, t. 1409, wrz. 2014.
- [22] J. Ni, T. Young, V. Pandelea, F. Xue i E. Cambria, *Recent Advances in Deep Learning Based Dialogue Systems: A Systematic Survey*, 2022. arXiv: 2105.04387 [cs.CL].
- [23] W. X. Zhao i in., *A Survey of Large Language Models*, 2023. arXiv: 2303.18223 [cs.CL].
- [24] J. Kaplan i in., *Scaling Laws for Neural Language Models*, 2020. arXiv: 2001.08361 [cs.LG].
- [25] M. Mateas i A. Stern, "Towards Integrating Plot and Character for Interactive Drama", sty. 2002.
- [26] M.-L. Ryan, "Avatars of Story". sty. 2006.
- [27] G. Gygax, *Player's Handbook*. Lake Geneva, WI: TSR Games, 1978, s. 9–15.
- [28] C. Crawford, "Chris Crawford on Interactive Storytelling / C. Crawford.", sty. 2005.
- [29] M. O. Riedl i V. Bulitko, "Interactive Narrative: An Intelligent Systems Approach", *AI Magazine*, t. 34, nr. 1, s. 67–77, 2013. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1609/aimag.v34i1.2449>. adr.: <https://onlinelibrary.wiley.com/doi/abs/10.1609/aimag.v34i1.2449>.
- [30] T. Short i T. Adams, *Procedural generation in game design*. CRC Press, czer. 2017.
- [31] M. Bernal Merino, *Translation and localisation in video games: Making entertainment software global*. sty. 2014, t. 6, s. 1–302.
- [32] K. Papineni, S. Roukos, T. Ward i W.-J. Zhu, "Bleu: a Method for Automatic Evaluation of Machine Translation", *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak i D. Lin, red., Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, lip. 2002, s. 311–318. adr.: <https://aclanthology.org/P02-1040>.
- [33] S. Banerjee i A. Lavie, "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments", *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, J. Goldstein, A. Lavie, C.-Y. Lin i C. Voss, red., Ann Arbor, Michigan: Association for Computational Linguistics, czer. 2005, s. 65–72. adr.: <https://aclanthology.org/W05-0909>.
- [34] T. Mikolov, K. Chen, G. S. Corrado i J. M. Dean, "Efficient estimation of word representations in vector space", *arXiv (Cornell University)*, sty. 2013. adr.: <http://export.arxiv.org/pdf/1301.3781.pdf>.
- [35] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin i J. Pineau, "How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation", *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh i X. Carreras, red., Austin, Texas: Association for Computational Linguistics, list. 2016, s. 2122–2132. adr.: <https://aclanthology.org/D16-1230>.
- [36] F. Jurčíček i in., "Real user evaluation of spoken dialogue systems using Amazon Mechanical Turk", *Proc. Interspeech 2011*, 2011, s. 3061–3064.
- [37] T. B. Brown i in., *Language Models are Few-Shot Learners*, 2020. arXiv: 2005.14165 [cs.CL]. adr.: <https://arxiv.org/abs/2005.14165>.

- [38] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE", *Mind*, t. LIX, nr. 236, s. 433–460, paź. 1950. eprint: <https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf>. adr.: <https://doi.org/10.1093/mind/LIX.236.433>.
- [39] P. R. Cohen, "If Not Turing's Test, Then What?", *AI Magazine*, t. 26, nr. 4, s. 61, grud. 2005. adr.: <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1849>.
- [40] J. van Stegeren i J. Myśliwiec, "Fine-Tuning GPT-2 on Annotated RPG Quests for NPC Dialogue Generation", *Proceedings of the 16th International Conference on the Foundations of Digital Games*, ser. FDG '21, Montreal, QC, Canada: Association for Computing Machinery, 2021. adr.: <https://doi.org/10.1145/3472538.3472595>.
- [41] T. Ashby, B. K. Webb, G. Knapp, J. Searle i N. Fulda, "Personalized Quest and Dialogue Generation in Role-Playing Games: A Knowledge Graph- and Language Model-Based Approach", *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23, Hamburg, Germany: Association for Computing Machinery, 2023. adr.: <https://doi.org/10.1145/3544548.3581441>.
- [42] Q. C. Gao i A. Emami, "The Turing Quest: Can Transformers Make Good NPCs?", *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, V. Padmakumar, G. Vallejo i Y. Fu, red., Toronto, Canada: Association for Computational Linguistics, lip. 2023, s. 93–103. adr.: <https://aclanthology.org/2023.acl-srw.17>.
- [43] OpenAI i in., *GPT-4 Technical Report*, 2024. arXiv: 2303.08774 [cs.CL]. adr.: <https://arxiv.org/abs/2303.08774>.
- [44] A. Radford, K. Narasimhan, T. Salimans i I. Sutskever, "Improving language understanding by generative pre-training", 2018.
- [45] Unity Technologies, *Creating and Using Scripts*, Uzyskano dostęp: 17/09/2024, 2020. adr.: <https://docs.unity3d.com/2020.1/Documentation/Manual/CreatingAndUsingScripts.html>.

SPIS RYSUNKÓW

2.1	Podział NLP	14
2.2	Model transformer [21]	18
5.1	Diagram architektury	41
5.2	Diagram pakietów	42
5.3	Diagram komponentów	45
5.4	Struktura projektu	47
5.5	Diagram przepływu mechaniki interakcji	49
5.6	Diagram przepływu dialogu	52
5.7	Zawartość komponentu Personality	53
5.8	Przebieg metody SendReply	56
5.9	Przykładowa mechanika - Przewodnik	60
5.10	Mechanika - przekazanie zadania	60
5.11	ScriptableObject QuestInfoSO	61
5.12	Cykl życia zadania	63
5.13	Animacje postaci gracza	68
5.14	Diagram sekwencji interfejsu użytkownika	70
5.15	Hierarchia sceny	71
5.16	Świat gry	73
5.17	Postać gracza	74
5.18	Postacie niezależne	74
5.19	Przeciwnicy	75
5.20	Panel startowy gry	75
5.21	Świat gry	76
5.22	Rejestr zadań	76
5.23	Menu	77
5.24	Modal przyjaciela	77
5.25	Modal wroga	78
5.26	Panel dialogowy	78
5.27	Panel wyboru dialogu	79
5.28	Panel wpisywania dialogu	79
5.29	Panel konwersji mowy na tekst	79
5.30	Źródło zbierania mowy	80
5.31	Panel odrodzenia	80
5.32	Ankieta	81
5.33	Organizacja zapisanych danych dialogowych	84
5.34	Struktura danych dialogu przechowywanych na serwerze Unity Cloud	84
5.35	Dane obecnych osiągnięć gracza i celów NPC	85
5.36	Przepływ danych	90
6.1	Wstęp do ankiety	93

6.2 Ankieta 1	93
6.3 Ankieta 2	93
6.4 Ankieta 3	93
6.5 Ankieta 4	94
6.6 Ankieta 5	94
6.7 Ankieta 6	94
6.8 Ankieta 7	94
6.9 Ankieta 8	94
6.10 Ankieta 9	94
6.11 Ankieta 10	95
6.12 Ankieta 11	95
6.13 Ankieta 12	95
6.14 Ankieta 13	95
6.15 Ankieta 14	95
6.16 Ankieta 15	95
6.17 Zakończenie ankiety	95
6.18 Wyniki pytań ogólnych typu q	98
6.19 Ocena interakcji z NPC - wyniki	100
6.20 Ocena jakości systemu - wyniki	101
6.21 Ocena dialogów - wyniki	102
6.22 Ocena klarowności gry - wyniki	102
6.23 Ocena satysfakcji i zaangażowania - wyniki	103
6.24 Ocena błędów i niespójności w dialogach - wyniki	104
6.25 Macierz korelacji	105

SPIS TABEL

6.1 Tabela pytań	97
6.2 Pytania do GPT	109