

## STRESZCZENIE

Celem pracy inżynierskiej jest projekt i realizacja prototypowej gry będącej hybrydą gatunków strategii czasu rzeczywistego oraz komputerowych gier fabularnych, osadzonej w realiach historycznych wybranej epoki. Niniejsza praca zawiera opis przykładowych rozwiązań zastosowanych w wybranych grach dostępnych na rynku oraz projekt i implementację wytwarzanego programu. Praca skupia się na przedstawieniu zaimplementowanych mechanik oraz sposobie oddania realiów, w których osadzona jest fabuła opracowanej gry.

Do przygotowania prototypu gry wykorzystano silnik Unity oraz udostępniane przez niego narzędzia. Za ich pomocą przygotowano podstawowe mechanizmy, typowe dla gier strategii czasu rzeczywistego i komputerowych gier fabularnych oraz przykładowe zadania tworzące fabułę gry. Utworzony prototyp pozwala na dalsze rozwijanie świata gry poprzez dodawanie nowych zadań oraz elementów rozgrywki.

W celu przygotowania się do pracy nad prototypem gry, przeprowadzono przegląd wybranych gatunków gier komputerowych oraz mechanik stosowanych w grach. Niniejsza praca zawiera opis działania poszczególnych mechanizmów na podstawie przykładów z gier dostępnych na rynku oraz przedstawienie ich wpływu na rozgrywkę. Przedstawione rozwiązania stanowią inspirację dla wytwarzanej gry.

W ramach projektu opracowano prototyp gry osadzonej w realiach historycznych wczesnego średniowiecza. W grze zostały zawarte mechanizmy zarządzania przyjaznymi jednostkami, budowania budowli oraz sztuczna inteligencja przeciwników, czyli mechaniki typowe dla gier RTS. Dodatkowo gra udostępnia prosty kontroler postaci, system rozwoju postaci oraz mechanizm interakcji z postaciami niezależnymi, które to są charakterystyczne dla gier cRPG.

Praca została zrealizowana przez trzech współautorów. Bartosz Strzelecki był odpowiedzialny za przygotowanie systemu dialogów, sztucznej inteligencji postaci niezależnych, mechanizmu widzenia przez przeskody oraz zapisu i odczytu stanu gry. Jest autorem podrozdziałów: 2.1.1, 2.2, 2.6, 2.11, 2.12, 3.2.2, 3.3.2, 4.1, 4.9, 4.11, 4.14, 4.10, 5.7, 5.9, 5.6, 5.5 i 5.10. Jest również współautorem podpunktów 2.1.4, 2.10, 5.11 oraz 6.3. Bogna Lew odpowiadała za modelowanie terenu gry oraz implementację mechanizmu budowania i kontrolera postaci. Jest autorką podrozdziałów 2.1.2, 2.3, 2.9, 2.4, 3.1, 3.1, 3.2.1, 3.3.1, 4.4, 4.3, 4.8, 4.12, 5.4, 5.8 i 6.1 oraz współtworzyła 1, 2.1.4, 2.10, oraz 5.11. Zofia Sosińska była odpowiedzialna za przygotowanie interfejsu użytkownika wraz z mechanizmem przetwarzania interakcji z użytkownikiem. Jest autorką podrozdziałów 2.1.3, 2.5, 2.7, 4.6, 4.5, 4.7, 4.13, 5.2, 5.1, 5.3 oraz 6.2. Współtworzyła również punkty 1 i 6.3.

## **ABSTRACT**

The purpose of the enginnering thesis is a project and a prototype of real-time strategy game set in historical reality of selected era. This work contains a description of sample solutions used in selected games available on the market and the design and a overview of the implementation of the produced program. The work focuses on the presentation of the implemented mechanics as well as the way of rendering the realities in which the plot of the developed game is set.

The Unity engine and the tools it provides were used to prepare a prototype of the game. Employing these tools, basic mechanics typical of real-time strategy games and computer role-playing games were prepared, as well as sample tasks forming the games's plot. The devised prototype allows for further development of the game environment by adding new activities and gameplay elements.

In preparation for the design and implementation of the game, a review of selected mechanics in games available on the market were conducted. This paper describes how they work using examples from existing games and how they affect their gameplay. The solutions presented provide inspiration for the developed game.

Within the project, a prototype game set in the historical realities of the early Middle Ages was developed. The game included mechanics of managing friendly units, building objects and artificial intelligence of the opponents, which are typical mechanics of RTS games. Additionally, the game facilitates simple character controller, character development system and a mechanic of interacting with non-playable characters, which are representative for cRPG games.

Diploma was carried out by three coauthors. Bartosz Strzelecki was responsible for preparing dialog system, artificial intelligence of the non-playable characters, vision through objects mechanic as well as saving and loading state of a game. He is the author of sections 2.1.1, 2.2, 2.6, 2.11, 2.12, 3.2.2, 3.3.2, 4.1, 4.9, 4.11, 4.14, 4.10, 5.7, 5.9, 5.6, 5.5 and 5.10. He also coauthored 2.1.4, 2.10, 5.11 and 6.3. Bogna Lew was responsible for modeling of the game's world as well as implementation of build mechanic and character controller. She is the author of sections 2.1.2, 2.3, 2.9, 2.4, 3.1, 3.1, 3.2.1, 3.3.1, 4.4, 4.3, 4.8, 4.12, 5.4, 5.8 and 6.1 and coauthored 1, 2.1.4, 2.10 and 5.11. Zofia Sosińska was responsible for preparing user interface with a system for resolving user input. She is the author of sections 2.1.3, 2.5, 2.7, 4.6, 4.5, 4.7, 4.13, 5.2, 5.1, 5.3 and 6.2. She also coauthored sections 1 and 6.3.

## **SPIS TREŚCI**

Wykaz ważniejszych oznaczeń i skrótów .....	6
1. Wstęp i cel pracy (Bogna Lew, Zofia Sosińska) .....	7
2. Przegląd wybranych technologii .....	9
2.1. Przegląd wybranych gatunków gier komputerowych .....	9
2.1.1. Gry RTS (Bartosz Strzelecki) .....	9
2.1.2. Gry cRPG (Bogna Lew) .....	10
2.1.3. Specyfika strategicznych gier turowych (Zofia Sosińska) .....	11
2.1.4. Porównanie wybranych gatunków gier. (Bogna Lew, Bartosz Strzelecki) .....	11
2.2. Model sztucznej inteligencji przeciwników w grach RTS (Bartosz Strzelecki) .....	12
2.3. Mechanizm budowania oraz zarządzanie zasobami w grach RTS (Bogna Lew) .....	13
2.4. Fabuła w grach RTS (Bogna Lew) .....	14
2.5. Interfejs użytkownika w grach (Zofia Sosińska) .....	15
2.6. System dialogów w grach (Bartosz Strzelecki) .....	17
2.7. Sterowanie jednostkami w grach (Zofia Sosińska) .....	19
2.8. Kompasy w grach (Zofia Sosińska) .....	20
2.9. Sterowanie postacią oraz walka (Bogna Lew) .....	20
2.10. Rozwój postaci gracza w grach (Bartosz Strzelecki, Bogna Lew) .....	21
2.11. Przekazywanie informacji o świecie w grach (Bartosz Strzelecki) .....	23
2.12. Modele celowania w grach (Bartosz Strzelecki) .....	25
3. Technologie, algorytmy i narzędzia .....	26
3.1. Przegląd silników gier (Bogna Lew) .....	26
3.2. Narzędzia dostępne dla silnika Unity .....	27
3.2.1. Terrain Toolbox (Bogna Lew) .....	27
3.2.2. Navmesh (Bartosz Strzelecki) .....	28
3.3. Struktury danych .....	28
3.3.1. ScriptableObject (Bogna Lew) .....	28
3.3.2. Protobuf (Bartosz Strzelecki) .....	29
4. Projekt systemu .....	30
4.1. Organizacja (Bartosz Strzelecki) .....	30
4.1.1. Główne etapy projektu .....	30
4.2. Skład zespołu projektowego .....	31
4.3. Analiza i specyfikacja wymagań (Bogna Lew, Zofia Sosińska) .....	31
4.3.1. Specyfika wymagań wynikających z założeń projektu .....	31
4.3.2. Wymagania funkcjonalne .....	31
4.3.3. Wymagania pozafunkcjonalne .....	32
4.3.4. Diagram przypadków użycia .....	32
4.3.5. Diagram stanów .....	33
4.3.6. Diagram klas .....	33

4.4. Opis świata gry i charakteru rozgrywki (Bogna Lew).....	34
4.5. Interfejs użytkownika (Zofia Sosińska) .....	35
4.5.1. Interfejs podstawowy .....	35
4.5.2. Menu stawiania budynków.....	35
4.5.3. Menu wydawania komend .....	37
4.5.4. Menu zapisu.....	37
4.5.5. Informacja o możliwej interakcji.....	38
4.5.6. Dziennik z zadaniami .....	38
4.5.7. Ekran końca gry .....	39
4.6. Menu główne (Zofia Sosińska).....	39
4.7. Nawigacja (Zofia Sosińska).....	40
4.8. Poruszanie postacią (Bogna Lew).....	40
4.9. Rozwój postaci gracza (Bartosz Strzelecki).....	41
4.10.Widzenie przez ściany (Bartosz Strzelecki) .....	41
4.11.System dialogów (Bartosz Strzelecki) .....	42
4.12.Mechanizm budowania (Bogna Lew).....	42
4.13.Sterowanie jednostkami, podążanie za główną postacią (Zofia Sosińska) .....	42
4.14.Sztuczna inteligencja (Bartosz Strzelecki) .....	43
5. Implementacja .....	44
5.1. Interfejs Użytkownika (Zofia Sosińska).....	44
5.1.1. Interfejs podstawowy .....	44
5.1.2. Menu stawiania budynków.....	45
5.1.3. Menu wydawania komend .....	45
5.1.4. Menu zapisu.....	46
5.1.5. Informacja o możliwej interakcji.....	46
5.1.6. Dziennik z zadaniami .....	48
5.1.7. Ekran końca gry .....	48
5.2. Menu główne (Zofia Sosińska).....	49
5.3. Zasady działania kompasu (Zofia Sosińska) .....	50
5.4. Kontroler postaci (Bogna Lew) .....	52
5.5. Rozwój postaci gracza (Bartosz Strzelecki) .....	54
5.6. Widzenie przez horyzont (Bartosz Strzelecki) .....	54
5.7. System dialogów (Bartosz Strzelecki) .....	56
5.8. Mechanizm budowania (Bogna Lew).....	57
5.9. Sztuczna inteligencja (Bartosz Strzelecki) .....	60
5.9.1. Mechanika zachowań klas jednostek.....	60
5.9.2. Sztuczna inteligencja przyjaznych jednostek.....	62
5.10.Zapis i odczyt stanu gry (Bartosz Strzelecki) .....	63
5.11.Przebieg rozgrywki (Bartosz Strzelecki, Bogna Lew) .....	64
6. Podsumowanie .....	69
6.1. Osiągnięte efekty (Bogna Lew) .....	69
6.2. Opinia osoby z zewnątrz (Zofia Sosińska) .....	69
6.3. Dalsze możliwości rozwoju (Bartosz Strzelecki, Zofia Sosińska) .....	70

Wykaz literatury.....	72
Wykaz rysunków.....	74
Wykaz tabel.....	75
Załącznik A.....	76

## WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

<b>RTS</b>	Gra strategii czasu rzeczywistego (ang. <i>real-time strategy</i> ). Jest to gatunek gier, w którym gracz nie jest ograniczany przez turowość i kolejność ruchów. Wymusza szybsze podejmowanie decyzji, a ich skutki są natychmiastowe.
<b>cRPG</b>	Komputerowa gra fabularna (ang. <i>computer role-playing game</i> ). Gatunek gier, w którym gracz wciela się w postać lub drużynę, przemieszczając się w świecie stworzonym przez autorów gry.
<b>TBS</b>	Strategiczna gra turowa (ang. <i>turn-based strategy</i> ). Jest to podgatunek gier strategicznych, w którym gracze wykonują swoje akcje w turach.
<b>AAA (Triple-A)</b>	Termin stosowany w przemyśle gier komputerowych. Służy do określenia wysoko-budżetowych gier, od których oczekuje się wysokiej jakości.
<b>Wielkie odkrycia geograficzne</b>	Termin odnoszący się do odkryć geograficznych, które miały miejsce na przełomie XV i XVI wieku.
<b>AI</b>	Sztuczna inteligencja (ang. <i>artificial intelligence</i> ) jest wykorzystywana do imitowania intelligentnego zachowania postaci niezależnych.
<b>NPC</b>	Termin określający postacie niezależne (ang. <i>non-playable character</i> ), czyli postacie, które nie są kontrolowane bezpośrednio przez gracza.
<b>Input Manager</b>	Komponent silnika Unity, który umożliwia definiowanie wirtualnych osi oraz przypisywanie do nich odpowiednich klawiszy. Poprzez odczyt wartości zwracanej przez oś możliwe jest wyznaczenie odpowiedniej akcji.
<b>Unity Runtime</b>	Biblioteka Unity, która implementuje funkcjonalności silnika poza edytorem.
<b>Asset Store</b>	Sklep stworzony przez Unity, w którym dostępne jest wiele zarówno darmowych, jak i płatnych zasobów, które można wykorzystać do tworzenia gier komputerowych.
<b>UI</b>	Interfejs użytkownika (ang. <i>user interface</i> ), czyli oprogramowanie umożliwiające użytkownikowi interakcję z systemem.
<b>Mapa wysokości</b>	(ang. <i>heightmap</i> ) Jest to termin określający zdjęcie w odcieniach szarości obejmujące wysokość terenu. Kolor czarny na zdjęciu reprezentuje najniższe punkty, natomiast kolor biały - najwyższe.
<b>Efekt Fresnela</b>	Efekt powstający w trakcie padania światła na granicę pomiędzy różnymi ośrodkami optycznymi, będący skutkiem odbicia i przepuszczenia światła.

## 1. WSTĘP I CEL PRACY (BOGNA LEW, ZOFIA SOSIŃSKA)

Postrzeganie świata przed erą wielkich odkryć geograficznych znaczaco różniło się od tego, które dominuje obecnie. Dawniej dowódcy nie podejmowali decyzji strategicznych na podstawie precyzyjnych map i pełnej wiedzy o aktualnym stanie zadania, lecz zmuszeni byli budować przestrzenny obraz istniejącej sytuacji w toku dyskusji z naocznymi świadkami, takimi podróżnicy. Dobrze obrazującym ówczesne postrzeganie przestrzeni przykładem jest mapa Imperium Rzymskiego, pokazana na rysunku 1.1. Czytanie jej dosłownie mija się z celem, gdyż nie są na niej zachowane ani proporcje, ani strony świata. Mimo tego, że basen Morza Śródziemnego został ówcześnie dosyć dokładnie oddany, "nie wydaje się, aby Rzymianom współczesna kartograficzna wierność była potrzebna" [1].



Rysunek 1.1: Mapa basenu Morza Śródziemnego z czasów Imperium Rzymskiego

Aspekt historyczny jest jednym z popularnych motywów w grach komputerowych. Obecnie istnieje wiele tytułów, których fabuła jest osadzona w realiach historycznych. O takich grach można powiedzieć, że są "formą publicznej dyskusji pomiędzy projektantem a graczami. Możemy uznać graczy za uczestników chcących wysłuchać argumentów stawianych przez projekt gry. To też może oznaczać, że postrzeganie czy rozumienie przeszłości przez uczestnika może być kształtowane lub zmieniane przez ukryty, lub wyraźny argument przedstawiany przez projekt gry" [2].

Współczesne gry komputerowe, których fabuła osadzona jest w realiach historycznych, stosuje wiele uproszczeń. Ma to na celu poprawienie jakości rozgrywki gracza, jednakże sprawia, że nie oddaje w pełni realiów. Często obraz dzisiejszych możliwości zasłania faktyczne dowodzenie w czasach, w których dzieje się gra. Technologia nieznana starożytnym teraz pozwala oficerom na wydawanie rozkazów na podstawie dokładnych map poprzez radio, jednak w dawniejszych czasach było to niemożliwe. W większości gier historycznych grywalna postać momentami ma wręcz boskie umiejętności i wiedzę. Zwykły człowiek nie był w stanie w jednej chwili zobaczyć całego świata i wydać komendę jednostkom, znajdującym się na

drugim jego końcu, o przemieszczeniu się do punktu z dokładnością do jednego metra.

Celem projektu jest zaprojektowanie oraz zaimplementowanie prototypowej gry strategii czasu rzeczywistego RTS (ang. *real-time strategy*) osadzonej we wczesnym średniowieczu z trybem rozgrywki dla jednego gracza. Gra będzie oferować również elementy typowe dla komputerowych gier fabularnych, co ma stanowić urozmaicenie rozgrywki. Udostępniane przez nią mechaniki mają jak najlepiej oddawać realia tamtych czasów, przy jednoczesnym zachowaniu podstawowych cech tego gatunku. Omówione rozwiązanie powinno zbalansować narzucone ograniczenia tak, aby nie pogarszać jakości rozgrywki.

W rozdziale *Wprowadzenie do dziedziny* zostały przedstawione przykłady podstawowych mechanik w grach strategii czasu rzeczywistego. Kolejny rozdział skupia się na podstawowych narzędziach oraz technologiach wykorzystywanych do wytwarzania gier komputerowych. Dodatkowo preczytuje wybrane przez zespół środowisko, które zostanie wykorzystane do implementacji rozwiązań. W kolejnym rozdziale został przedstawiony projekt wytwarzanej gry. Skupia się on na wizji zespołu oraz oczekiwanych efektach. Na koniec, w rozdziale *Implementacja* zaprezentowane zostały osiągnięte rezultaty, w jaki sposób je opracowano oraz efekt końcowy projektu.

## 2. PRZEGŁĄD WYBRANYCH TECHNOLOGII

W niniejszym rozdziale zostały przedstawione wybrane gatunki gier komputerowych oraz mechaniki wykorzystywane w grach, których działanie zostało omówione na podstawie przykładów z wybranych gier. Celem tego jest obrazowe zaprezentowanie ich integralności w grach oraz wpływu na rozgrywkę.

Czynnikami, które mają duży wpływ na odbiór gry są m.in. interfejs użytkownika oraz mechanizmy interakcji. Twórcy bardzo często stosują w nich uproszczenia, co ma na celu ułatwienie graczy wykonywanie zadań. Ma to jednak ogromny wpływ na zachowanie realizmu w grach, zwłaszcza tych opartych na wydarzeniach historycznych oraz prawdziwym świecie.

### 2.1. Przegląd wybranych gatunków gier komputerowych

W niniejszym podrozdziale zostały przedstawione wybrane gatunki gier komputerowych, które są najistotniejsze z punktu widzenia projektu. Przede wszystkim zostały opisane gry typu strategii czasu rzeczywistego oraz strategiczne gry turowe TBS (ang. *turn-based strategy*). Ma to na celu zaprezentowanie podstawowych różnic pomiędzy tymi gatunkami.

Podział gier na gatunki pozwala na określenie podstawowych cech gry i daje ogólne pojęcie o jej charakterze. Celem takiego podziału gier komputerowych jest przede wszystkim pokazanie użytkownikowi jakich mechanik może się spodziewać i jak będzie wyglądać przebieg rozgrywki. Można więc powiedzieć, że "w grach gatunek niesie więcej informacji niż temat i sceneria" [3]. Z tego powodu aby zrozumieć cel projektu ważne jest zauważenie podobieństw i różnic pomiędzy kluczowymi dla niego gatunkami.

#### 2.1.1. Gry RTS (Bartosz Strzelecki)

Strategie czasu rzeczywistego RTS (ang. *real-time strategy*) odróżnia od innych gatunków występowanie zarówno strategicznych, jak i taktycznych elementów rozgrywki, w której czas, w tym szybkość reakcji ma niezwykle istotne znaczenie. Podstawowe założenia klasycznych gier RTS to skomplikowana równowaga pomiędzy zarządzaniem zasobami, budowaniem budynków oraz dowodzeniem armii. Centralnym elementem rozgrywki RTS jest zarządzanie różnorodnymi jednostkami, z których każda posiada unikalne zdolności i rolę na polu bitwy.

Gry te działają w dynamicznym środowisku czasu rzeczywistego, co odróżnia je od strategii turowych TBS (ang. *turn-based strategy*). Ten model wymusza szybkie podejmowanie decyzji i wymaga od gracza podzielności uwagi. Strategie czasu rzeczywistego zwykle też posiadają bardziej złożoną konstrukcję mapy, zawierającą skomplikowane rozłożenie celów i przeszkód, natomiast mapy w strategiach turowych zazwyczaj wykorzystują ruch oparty o siatkę, co zapewnia dużo bardziej zorganizowane środowisko rozgrywki.

W grach strategicznych czasu rzeczywistego w trybie kampanii zachowanie przeciwników jest zaprojektowane z myślą o zanurzeniu gracza w fabularnej opowieści, jednocześnie prezentując wyzwania związane z rozgrywką. Akcje wykonywane przez sztuczną inteligencję są dostosowane do celów danej misji, co pozwala na dopasowanie do obowiązującej narracji.

Tego typu produkcje pozwalają również na wciągającą rozgrywkę w trybie wieloosobowym, w którym gracze mogą rywalizować między sobą, jak również współpracować w celu pokonania wspólnego przeciwnika kontrolowanego przez sztuczną inteligencję. Na ten styl rozgrywki jest nakładany istotny nacisk

w wielu grach tego gatunku, poprzez zapewnienie zróżnicowanych grywalnych frakcji, zachowując przy tym symetrię balansu rozgrywki.

Z uwagi na cechę gier gatunku RTS polegającej na ciągłym przepływie czasu, gracze są zmuszeni do sprawnego reagowania na ciągle rozwijające się wydarzenia. Mijający czas wywiera presję na użytkowników, wymagając natychmiastowej reakcji w podejmowanych decyzjach. Z tego powodu w grach z tego gatunku efektywne zarządzanie czasem jest kluczowym elementem wymaganym do osiągnięcia oczekiwanych rezultatów. Zazwyczaj czas upływa w sposób jednostajny, co oznacza, że nie mogą spowolnić lub zatrzymać tempa rozgrywki, aby odetchnąć lub przemyśleć swój plan działania.

Ogólnie rzecz biorąc, gatunek RTS oddaje dreszcz emocji związanych z prowadzeniem działań strategicznych, co wymaga połączenia zaradności, umiejętności przewidywania oraz sprytnego podejmowania decyzji na szybkim i stale zmieniającym się polu bitwy. "Wewnątrz tej ogólnej definicji znajdują się pewne podgatunki. [...] Tradycyjne gry RTS składają się z elementów budowania bazy, zarządzania zasobami i drzew technologicznych. Do tej kategorii pasują gry takie jak *Command & Conquer*, *StarCraft* i *Age of Empires*." [4]. Do gatunku gier strategii czasu rzeczywistego należą również takie tytuły jak *Company of Heroes*<sup>1</sup> kanadyjskiego studia Relic Entertainment wyprodukowane w 2013 roku. Producent jest znany również z gry *Warhammer 40,000: Dawn of War*<sup>2</sup> powstałego w 2006 roku. Natomiast najbardziej przełomową grą tego gatunku jest gra *Starcraft*<sup>3</sup> wydana przez Blizzard Entertainment w 1998 roku, jest to jedna z najlepiej sprzedających się gier w historii gatunku [5].

### 2.1.2. Gry cRPG (Bogna Lew)

Komputerowe gry fabularne cRPG (ang. *computer role-playing game*) czerpią inspirację z tradycyjnych gier fabularnych. Komputerowa gra fabularna opowiada pewną historię, w której gracz wciela się w wybraną postać bądź drużynę i za ich pomocą eksploruje świat gry, wykonując zadania oraz rozwijając łamigłówki. Gry tego typu cechują się zwykle nieliniową, z góry zdefiniowaną fabułą. Czasami jest ona podzielona na rozdziały, które gracz musi kolejno ukończyć, aby móc kontynuować rozgrywkę.

Istotnym elementem gier fabularnych jest rozwój postaci. W trakcie rozgrywki gracz może kreować swojego bohatera poprzez podnoszenie jego współczynników, dodawanie mu nowych umiejętności, czy zmienianie ekwipunku. Gra umożliwia użytkownikowi zmianę statystyk swojej postaci za każdym razem, gdy osiągnie następny poziom doświadczenia. Zwyczajowo punkty doświadczenia są graczowi przyznawane po wykonaniu kolejnych zadań bądź pokonaniu przeciwników.

Komputerowe gry fabularne zwykle cechują się otwartym światem, czasem zawierającym niewielkie ograniczenia, co umożliwia graczowi swobodne eksplorowanie świata. Aby ułatwić użytkownikowi nawigację i podróżowanie, komputerowe gry fabularne udostępniają mu system map, który pokazuje lokalizację głównych elementów świata. W trakcie swojej wędrówki gracz może wchodzić w interakcję z postaciami niezależnymi, od których może dostać zadania do zrealizowania. Może je wykonywać na różne sposoby, dzięki czemu może wpływać na fabułę gry.

Ważnym aspektem gier fabularnych są przeciwnicy, z którymi gracz może walczyć. W trakcie potyczki może wykorzystywać specjalne umiejętności swojej postaci, wykonywać ataki bądź przemieszczać się. W zależności od przyjętej przez twórców formy, walka może mieć charakter turowy lub przebiegać w czasie rzeczywistym. Pokonanie przeciwników przynosi pewne korzyści w postaci zdobywania punktów doświadczenia bądź łupu.

Do gatunku komputerowych gier fabularnych należą między innymi takie tytuły jak *Divinity: Original*

---

<sup>1</sup><https://www.companyofheroes.com>

<sup>2</sup><https://www.dawnofwar.com>

<sup>3</sup><https://starcraft.blizzard.com>

*Sin II*<sup>4</sup> wydane w 2017 roku przez Larian Studios, wyprodukowane przez studio CD Project Red w 2015 *Wiedźmin 3: Dziki Gon*<sup>5</sup>, czy gra *The Elder Scrolls V: Skyrim*<sup>6</sup> studia Bethesda Softworks wydane w 2016 roku.

#### 2.1.3. Specyfika strategicznych gier turowych (Zofia Sosińska)

Strategiczne gry turowe TBS (ang. *turn-based strategy*) są jednym z wielu popularnych gatunków gier komputerowych, ale w swej podstawowej mechanice działania są silnie zbliżone do gier planszowych. Te bardzo często trzymają się określonego schematu: konkretny gracz wykonuje jeden z dostępnych ruchów, a jego współgracze patrzą i czekają na swoją kolej. Zabronione jest wtedy przerywanie, czy przeszkadzanie mu i ma pełną autonomię podjęcia decyzji. Gdy dana osoba zakończy swój ruch, następny w kolejce gracz dostaje swoją szansę. Opisany schemat nosi nazwę "tury" i jest to, inaczej mówiąc, mechanika wymuszająca na graczech skolejkowanie się, przyznając każdemu po kolei prawo do podjęcia ściśle ustalonych akcji. Strategiczne gry turowe korzystają z tego schematu pełniąc rolę semafora, który zczytuje ruchy tylko jednego gracza.

Oprawą fabularną takich gier często jest jakaś forma bitwy. W wersji jednoosobowej użytkownik może być dowódcą pewnej drużyny, która napotykać będzie przeciwników. Po rozpoczęciu walki postacie ustawiane są w kolejkę. Sortowanie może odbywać się względem wylosowanych wartości, lub chociażby według umiejętności, takich jak np. szybkości. Każda postać dostaje swoją kolej na wykonanie konkretnej liczby ściśle określonych ruchów. To jak wiele może zostać podjętych jest ewaluowane według przyznanych im punktów trudności, tego ile postać ma energii, albo po prostu jest to z góry ustalone na przykładowo jeden. Pośród tur znajdują się też te przeciwników, których działaniami będzie kierować sztuczna inteligencja. W wersji wieloosobowej tylko jeden wykonuje ruch, a reszta ma zamrożony stan gry, obserwując decyzje współgracza.

Gry tego typu często wspierają rozwój drużyny i umiejętności, dając użytkownikowi dostęp do coraz to nowych mechanik. Może on często wybierać specjalizacje postaci tak, aby pokrywały się z jego taktyką. Sedno strategii tych gier leży w odpowiednim przyznaniu umiejętności, a następnie wykorzystaniu ich podczas walki w jak najoptymalniejszy sposób.

Do gatunku strategicznych gier turowych należą m.in. takie tytuły jak *Heroes of Might and Magic III HD Edition*<sup>7</sup> z 2015 wyprodukowana przez DotEmu, *Civilization VI: Gathering Storm*<sup>8</sup> z 2019 roku studia Firaxis Games oraz *Total War: Warhammer*<sup>9</sup> wydana przez Strategic Simulations, Inc. w 2016 roku.

#### 2.1.4. Porównanie wybranych gatunków gier. (Bogna Lew, Bartosz Strzelecki)

Tabela 2.1 przedstawia porównanie najważniejszych cech wymienionych wcześniej gatunków gier. Do najważniejszych elementów porównania należą między innymi turowość rozgrywki, typ fabuły oraz punkt widzenia gracza.

---

<sup>4</sup><https://divinity.game/>

<sup>5</sup><https://www.thewitcher.com/us/pl/witcher3>

<sup>6</sup><https://elderscrolls.bethesda.net/pl>

<sup>7</sup><https://www.ubisoft.com/pl-pl/game/heroes-of-might-and-magic-3-hd>

<sup>8</sup><https://civilization.com/pl-PL/civilization-6-gathering-storm/>

<sup>9</sup><https://www.totalwar.com/games/warhammer/>

**Tabela 2.1:** Porównanie gatunków gier.

Aspekt	RTS	RPG	TBS
Punkt widzenia gracza	Widok z lotu ptaka	Widok zza postaci	Widok z lotu ptaka
Turowy przebieg rozgrywki	Nie	Nie zawsze	Tak
Fabuła	Liniowe scenariusze	Zwykle nieliniowa	Losowo generowane warunki startowe
Świat	Ograniczony	Główne otwarty	Ograniczony
Styl rozgrywki	Główne wieloosobowy	Jednoosobowy	Wieloosobowy i jednoosobowy
Mapa	Gęsta siatka	Brak siatki	Rzadka siatka

## 2.2. Model sztucznej inteligencji przeciwników w grach RTS (Bartosz Strzelecki)

W większości gier, w tym omówionych w poprzednim podrozdziale, kluczową rolę odgrywa sztuczna inteligencja AI (ang. *artificial intelligence*), wspierając doświadczenia z rozgrywki. W tym przypadku termin odnosi się do zbioru algorytmów i systemów zaprojektowanych w celu symulacji zachowań podobnych do tych gracza, czyli między innymi umiejętności podejmowania decyzji i rozwiązywania problemów.

Sztuczna inteligencja przeciwników w grach takich jak Warcraft III<sup>10</sup> lub StarCraft II<sup>11</sup>, przede wszystkim w trybie kampanii, jest odpowiedzialna za kontrolowanie wrogich jednostek w celu zaoferowania graczowi wyzwania. Głównym zadaniem AI jest zasymulowanie strategicznych decyzji i wydajne zarządzanie zasobami. "W przypadku gier strategicznych czasu rzeczywistego ruch idzie w parze z odnajdywaniem ścieżki. Jednostki z pewnością potrzebują planu (np. ścieżki), w jaki sposób przedostanie się z jednej strony miasta na drugą." [6]. AI podejmuje decyzję na podstawie predefiniowanych zasad i algorytmów. Analizuje sytuację, w której się znajduje, biorąc pod uwagę siłę swojej własnej armii, siłę armii gracza oraz specjalne zdolności jednostek i środowisko, w którym toczy się gra. Ta analiza pozwala komputerowi na podejmowanie strategicznych decyzji, takich jak na przykład, kiedy atakować, bronić się, eksplorować, czy rozszerzać swoje terytorium. W tych grach sztuczna inteligencja może przybrać jeden z kilku wariantów wynikających z poziomu trudności. Wyższe poziomy dają przeciwnikowi przewagę taką jak wydajniejsze zbieranie zasobów lub szybszą produkcję jednostek.

Początkowo przeciwnik konstruuje i rozbudowuje swoją bazę, w celu zgromadzenia odpowiedniej liczby zasobów, szkolenia jednostek i prowadzenia badań. AI strategicznie rozmieszcza budynki i struktury obronne, aby ochronić swoją fortę przed najazdami gracza. Misje kampanii często też zawierają oskryptowane wydarzenia lub walki, które dodają głębi rozgrywce. Podczas tych starć wroga sztuczna inteligencja może zachowywać się w specjalny sposób, kontrolując potężne jednostki, do których gracz normalnie nie ma dostępu lub inicjując działania, które popychają narrację do przodu. Zachowanie wroga w kampanii jest zróżnicowane i obejmuje różnorodne cele misji i scenariusze. Gracze mogą napotkać wrogów, którzy preferują agresywne ataki, skupią się na strategiach obronnych lub specjalizują się w taktyce hit and run. Sztuczna inteligencja dostosowuje proces podejmowania decyzji do konkretnych wymagań misji, często wykorzystując ukształtowanie terenu, synergię jednostek i scenariusze wydarzeń, aby rzucić wyzwanie umiejętnościom gracza. Motywuje to do wykorzystywania myślenia strategiczne-

<sup>10</sup><https://warcraft3.blizzard.com>

<sup>11</sup><https://starcraft2.blizzard.com>

go, zarządzania zasobami i efektywnego dobierania jednostek, aby przezwyciężyć różnorodne strategie stosowane przez wrogą sztuczną inteligencję.

Ten model jest zaimplementowany między innymi przy użyciu algorytmów takich jak A\*, który jest rozszerzoną wersją algorytmu Dijkstry o wykorzystanie heurystyki optymalizującej wyszukiwanie. Stosowany jest również algorytm drzewa decyzyjnego pozwalający na zdefiniowanie zachowań agentów sztucznej inteligencji z zastosowaniem wydarzeń losowych.

### **2.3. Mechanizm budowania oraz zarządzanie zasobami w grach RTS (Bogna Lew)**

Jednym z typowych elementów gier strategii czasu rzeczywistego jest tworzenie baz i budowanie fortyfikacji. Mechanizm ten stanowi urozmaicenie rozgrywki i wprowadza dodatkowe aspekty możliwe do uwzględnienia w planowaniu strategii. Dla wielu gier RTS jest wręcz nieodłącznym elementem, który umożliwia graczowi tworzenie i rozwój nowych jednostek, produkcję zasobów, umacnianie swojej pozycji oraz zwiększanie swojej potęgi.

Mechanizm ten wiąże się z szeregiem ograniczeń, które mają kluczowy wpływ na rozgrywkę. Należą do nich między innymi ograniczenia związane z ukształtowaniem terenu oraz obecnością innych elementów scenerii. Każde z tych ograniczeń ma swoje źródło w prawdziwym świecie i mechanizm budowania musi je uwzględniać.

Z tą mechaniką związany jest system zasobów, który jest popularnym aspekiem gier z tego gatunku. Wiele gier strategii czasu rzeczywistego umożliwia graczowi budowanie własnej ekonomii. Uzyskane przez niego zasoby często mogą zostać wykorzystane przez mechanizm budowania jako koszta budowy obiektów.

Przykładem gry strategii czasu rzeczywistego implementującej tę mechanikę jest *Warhammer 40,000: Dawn of War*<sup>12</sup>. Jest to gra, której realia są osadzone w uniwersum gry bitowej *Warhammer 40,000*. Udostępnia ona tryb jednoosobowy oraz wieloosobowy dla maksymalnie sześciu graczy. W pierwszym wariantie gracz wciela się w postać dowódcy armii Space Marines z Blood Ravens i ma za zadanie zapobiec inwazji Orków. Gra *Warhammer 40,000: Dawn of War* bardzo szybko zyskała na popularności i oferowała wszystko, co było potrzebne dla tego gatunku. Z tego powodu warto się jej przyjrzeć, pomimo faktu, że jej realia znaczco odbiegających od tych, w których zostanie osadzona tworzona przez nas gra.

*Warhammer 40,000: Dawn of War* wyróżnia model pozyskiwania surowców. W grze dostępne są dwa rodzaje: Energia, która jest generowana przez dedykowane do tego budowle oraz Rekwizycja, której szybkość wytwarzania jest uzależniona od kontrolowanych przez gracza punktów strategicznych. Taka mechanika znacznie lepiej wpasowuje się w realia gry oraz wymusza na użytkowniku przyjęcie agresywniejszej strategii.

Dodatkowo *Warhammer 40,000: Dawn of War* posiada typowy dla gier RTS mechanizm tworzenia budowli. Gracz ma do dyspozycji jednostki, którym może zlecić budowę wybranego przez siebie obiektu po poniesieniu kosztów jego utworzenia. Zanim będzie możliwe rozpoczęcie budowania użytkownik musi wybrać miejsce, w którym budynek powstanie, co robi, przesuwając jego podgląd po mapie. W tym czasie gra dokonuje walidacji miejsca i informuje gracza czy wybrany obszar jest poprawny, odpowiednio podświetlając widok budynku. Wybudowanie obiektu nie jest natychmiastowe, co sprawia, że gra lepiej oddaje realia, w których jest osadzona.

---

<sup>12</sup><https://www.dawnofwar.com/>



Rysunek 2.1: Budowanie budynku przez dedykowaną do tego jednostkę w grze *Warhammer 40,000: Dawn of War*.

#### 2.4. *Fabuła w grach RTS (Bogna Lew)*

W przypadku wielu gier, niezależnie od gatunku, kluczową rolę odgrywa fabuła. To ona sprawia, że gra przestaje być jedynie programem wykonującym polecenia, a staje się opowieścią będącą formą rozrywki. Ważnym aspektem fabuły jest jej narracja, która definiuje kolejność występowania konkretnych elementów. W odróżnieniu od narracji budującej fabułę książki, czy filmu, narracja w grach komputerowych umożliwia graczowi wpływanie na nią. "Dobrze nakreślona narracja wykorzystuje wybory fabularne z wielkim uwzględnieniem ogólnego celu pracy. Może obejmować ujawnienie pewnych informacji z przeszłości, aby zwiększyć poczucie niepewności, lub wczesne informacje podstawowe, aby wzmacnić motywację postaci" [7].

W grach komputerowych w skład fabuły wchodzą cel główny oraz zadania poboczne. Cel główny stanowi podstawę gry, dając graczy poczucie dążenia do czegoś i jest ogólnym zamysłem całej rozgrywki. Zadania poboczne natomiast są krótkimi misjami będącymi urozmaiceniem rozgrywki. "W odniesieniu do gier komputerowych można więc mówić o podwójnym motywowaniu ich użytkowników, które dokonuje się na dwóch narracyjnych poziomach: jedna z motywacji wyznacza cel całej rozgrywce, druga natomiast jest ulokowana w przestrzeni pojedynczej misji i kończy wraz z jej zakończeniem" [8].

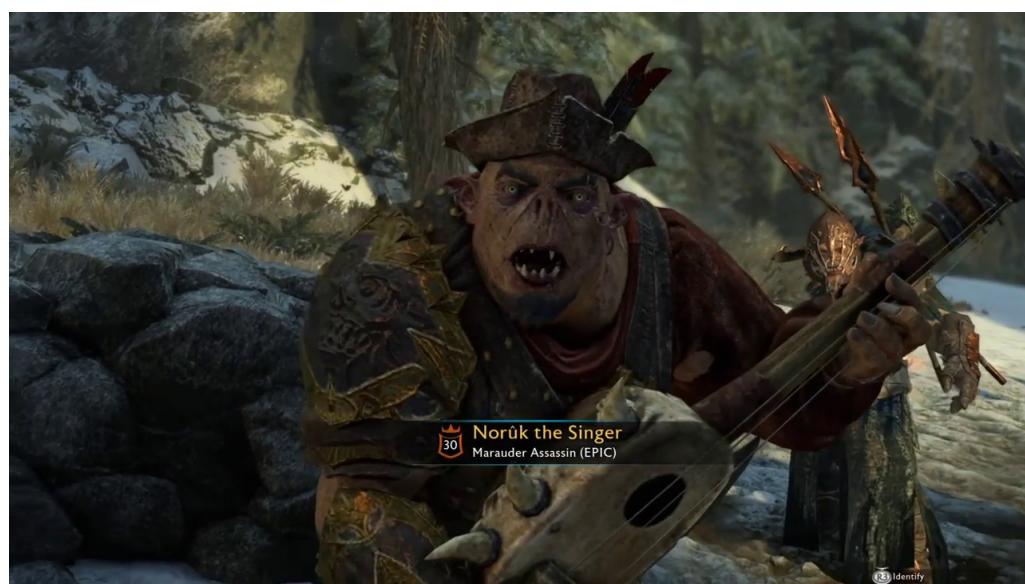
Innym elementem budującym fabułę są przerywniki filmowe, które wprowadzają nowe informacje i wydarzenia. "Wiele gier z wyrafinowaną fabułą wykorzystuje tę technikę do umiejscowienia działań gracza w fikcyjnym świecie, który dzięki temu może zostać opisany z dużą kontrolą po stronie autora." [9]. Stanowią one ciekawe urozmaicenie w rozgrywce i pozwalają na przekazanie graczowi więcej informacji.

W przypadku gier strategii czasu rzeczywistego fabuła występuje w trybie kampanii tych gier. Każda z pojedynczych bitew stanowi fragment większej historii występującej w kampanii. Typowo w grach typu RTS występuje fabuła liniowa. Przykładowo w grze *Warhammer 40,000: Dawn of War* ogólnym zamysłem kampanii jest obrona planety przed najazdem wroga. Zrealizowana jest za pomocą jednego nastu misji, w których gracz ma do realizacji pewne cele, takie jak zajęcie określonej liczby punktów

kontrolnych, czy zniszczenie bazy przeciwnika. Pomiędzy misjami występują przerywniki filmowe, które to odkrywają przed graczem kolejne elementy historii.

W tym miejscu warto wspomnieć System Nemesis występujący w grze *Middle-earth: Shadow of Mordor* i rozwinięty w *Middle-earth: Shadow of War*<sup>13</sup>. Chociaż wymienione gry nie są z gatunku strategii czasu rzeczywistego to ta mechanika jest interesującym przykładem kształtowania fabuły w grze. System Nemesis jest mechanizmem generowania przeciwników, który nadaje każdemu z nich ich unikalny charakter oraz wygląd. W ciągu gry system Nemesis dynamicznie wpływa na postacie, rozwijając je i odpowiednio zmieniając ich wygląd. Efektem tego jest płynnie zmieniająca się fabuła, która dla każdego gracza będzie inna.

Do podstawowych elementów tej mechaniki należy rozbudowany system relacji pomiędzy postaciami w grze, w tym postacią gracza. Dodatkowo buduje ona hierarchię wśród Orków, dynamicznie ją aktualizując w trakcie gry w zależności od śmierci poszczególnych bohaterów oraz postaci gracza.



Rysunek 2.2: Przykładowa postać przeciwnika utworzona przez system Nemesis.

System Nemesis jest interesującą mechaniką budującą fabułę w grze. Pozwala na zbudowanie unikatowych przeciwników, którzy mają bezpośredni wpływ na rozgrywkę i kształtowanie jej przebiegu.

## 2.5. Interfejs użytkownika w grach (Zofia Sosińska)

Dopuszczalnym założeniem przy tworzeniu gier komputerowych jest, że postać gracza ma jakąś wiedzę o świecie, w którym się znajduje oraz poziom inteligencji i umiejętności, które pozwalają jej przeanalizować konkretną sytuację i wyciągnąć wnioski. Skoro więc te informacje są w zasięgu możliwości wnioskowania postaci, gra ma prawo wyświetlić je użytkownikowi.

Takie graficzne przedstawienie najważniejszych danych z otoczenia gracza nazywamy interfejsem użytkownika UI (ang. *user interface*). Narzędziami odzwierciedlania informacji w czytelny i przystępny sposób mogą być takie elementy jak obrazki, teksty, czy wskaźniki. Takie zabiegi estetyczne sprawiają, że przetwarzamy suche dane na interesujące informacje. "Interfejs użytkownika w grze ma służyć zarówno rozrywce, jak i ułatwieniu dostępu." [10]. Dzięki UI możliwy jest wgląd w aktualny stan wiedzy postaci gracza, a tym samym lepsze zrozumienie otoczenia. Wymagające podkreślenia jest, że "[...] na stopień

<sup>13</sup><https://www.shadowofwar.com/>

zaangażowania gracza na poziomie świata gry duży wpływ ma interfejs, którym się posługuje. Mówiąc najprościej: gracz, który rozpoczyna grę, zanim dozna poczucia integracji ze sterowaną przez siebie postacią, musi - używając stwierdzenia Aarsetha - spełnić oczekiwania interfejsu." [8].

Jednym z kluczowych problemów przy projektowaniu elementów interfejsu użytkownika jest odpowiednie wybranie miejsca na ekranie. Konieczne jest zwrócenie uwagi na takie problemy, jak rozmiar, rola, priorytet dostarczenia informacji, częstość i długość występowania oraz częstotliwość zmiany wartości. Inną ważną cechą dobrze zaprojektowanego interfejsu użytkownika jest zbalansowanie między pokazaniem jak największej liczby kluczowych informacji, jednocześnie odkrywając je tak, aby nie przetłaczały gracza i nie zaciemniały przekazu. Często dane segreguje się według ich tematyki, ale czasem właściwsze może okazać się zgrupowanie ich względem tego, jak często gracz będzie ich potrzebował. Jeśli są to informacje niezbędne do zrozumienia aktualnego stanu świata gry, wartościowe może być pokazanie ich obok siebie. Wszystkie te czynniki mają za zadanie ułatwić użytkownikowi wyszukanie informacji. "Gdy czegoś szukamy, nasze spojrzenie jest prowadzone przez zainteresowanie wizualne. Trudne do wyobrażenia jest jak zainteresowanie wizualne działa; jakie rodzaje procesów działają w mózgu kiedy decydujemy się spojrzeć i na którym obszarze po chwili skupimy wzrok." [11]. Przy projektowaniu interfejsu użytkownika kluczowe jest rozplanowanie tego, które elementy, w jaki sposób i jak bardzo będą przykuwać uwagę użytkownika. Przy elementach, których treść użytkownik może przeglądać lub zmieniać najważniejszą cechą jest łatwość i przyjemność obsługi. "[...] prawdopodobnie nie ma innego wyjścia, niż projektowanie programów tak, aby nie potrzebowaly instrukcji obsługi." [12]. Inaczej będzie on gracza męczyć i rozpraszać.

Przykładem zaprojektowania zbalansowanego elementu interfejsu użytkownika jest rozwiążanie gry *Warcraft III: Reign of Chaos* studia Blizzard Entertainment. Zebrała ona najbardziej podstawowe informacje o świecie gry i skupiła te dane w wąskim pasku na samej górze ekranu. Skład elementów tej części interfejsu użytkownika jest niezmienny: pola otwierające zakładki, pora dnia oraz trzy wskaźniki zasobów. Pasek jest widoczny podczas całej rozgrywki, niezależnie od wykonywanych czynności. W tym statycznie zakotwiczonym na górze ekranu elemencie, dynamicznie zmieniają się jedynie ciągle aktualizowane informacje. Odpowiednio podmieniana jest tekstura pory dnia, zmieniająca się ze Słońca na Księżyc oraz stan zasobów, zależnie od wydania, czy pozyskania.



Rysunek 2.3: Lewa część paska z informacjami w grze *Warcraft 3*.



Rysunek 2.4: Prawa część paska z informacjami w grze *Warcraft 3*.

Innym przykładem dobrze zaplanowanego UI jest przedstawienie dostępnych do zbudowania pułapek w grze *Orcs must die!*<sup>14</sup> studia Robot Entertainment. Zadaniem gracza jest wcielenie się w jednego z Wojennych Magów i mordowanie nadciągających grup orków za pomocą różnorodnych broni i mechanizmów, które może postawić. Sam zakup mechanizmów jest sytuacją epizodyczną, dlatego grafiki pojawiają się, gdy gra jest w trybie stawiania budynków, i znikają poza nim. Takie rozwiązanie sprawia,

<sup>14</sup><https://robotentertainment.com/omd-details>

że użytkownik nie jest rozpraszaný przez niepotrzebny interfejs, gdy odpiera natarcie przeciwników. Zadbano także o optymalne umiejscowienie. Grafiki, będące miniaturami pułapek razem z ich ceną, pojawiają się na dole ekranu tak, aby nie zasłaniać potencjalnego miejsca do postawienia budowli.



Rysunek 2.5: Wyświetlenie dostępnych pułapek w grze *Orcs must die!*.

## 2.6. System dialogów w grach (Bartosz Strzelecki)

Systemy dialogów w grach wideo kształtują wciągającą historię, umożliwiając graczom dokonywanie wyborów, które wpływają na relacje między postaciami, zadania i narrację gry. Odkrywają wiedzę, pogłębiają zaangażowanie i oferują dynamiczną rozgrywkę poprzez różnorodne podejmowanie decyzji. Dialogi umożliwiają graczyowi wpłynięcie na świat, pozwalając mu wybrać, w którą stronę historia będzie podążać. Gracz w ten sposób rozwiązuje dylematy moralne i może wczuć się w klimat rozgrywki. "Najpopularniejsze zachodnie gry RPG, takie jak serie Baldur's Gate i Fallout, żyją i umierają dzięki sile dialogów i zdolności gracza do wpływania na postacie niezależne." [13].

W 'Mass Effect 3'<sup>15</sup> system dialogowy jest integralną częścią rozgrywki i pozwala graczom na prowadzenie rozmów z różnymi postaciami w trakcie gry. System dialogów w 'Mass Effect 3' wykorzystuje interfejs oparty na kole dialogowym (rys. 2.6), które przedstawia graczom wiele opcji odpowiedzi podczas rozmów, zwykle podzielonych na kategorie według ich ogólnego tonu lub intencji. Dostępne opcje często obejmują wybory dyplomatyczne, agresywne bądź konfrontacyjne oraz opcje neutralne lub śledcze. Podczas niektórych rozmów lub przerywników filmowych gracze mogą przerwać trwającą rozmowę, szybko wybierając określoną opcję dialogową. Te opcje przerywania pozwalają graczom podjąć natychmiastowe działania lub podjąć decyzje na miejscu, często wpływając na wynik sytuacji lub relacje postaci z innymi. Ogólnie rzecz biorąc, system dialogowy w 'Mass Effect 3' został zaprojektowany tak, aby zapewnić graczom bogate i wciągające doświadczenie w opowiadaniu historii, pozwalając im kształtować narrację poprzez wybory i interakcje z olbrzymią gamą postaci. System oferuje różnorodne opcje odpowiedzi, dynamiczne rozmowy i konsekwencje, przyczyniając się do fascynującej i rozgałęzionej narracji gry.

Alternatywnym rozwiązaniem jest to zaprezentowane w grze 'Fallout 3'<sup>16</sup> (rys. 2.7), które odróżniają

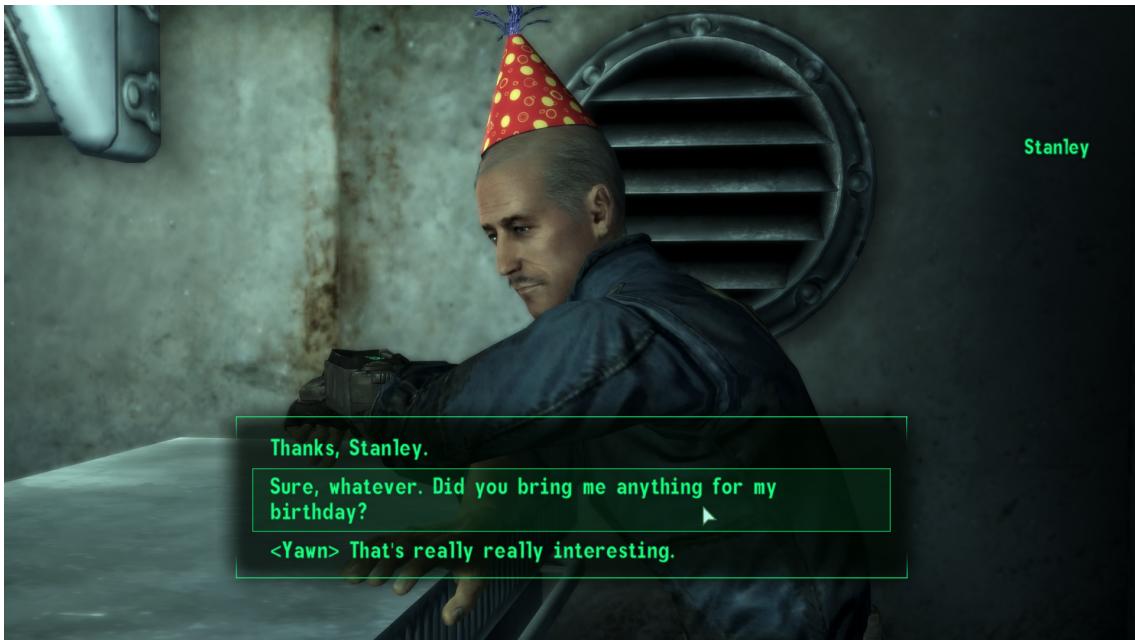
<sup>15</sup><https://www.ea.com/games/mass-effect/mass-effect-3>

<sup>16</sup><https://fallout.bethesda.net/pl>

przed wszystkim możliwe odpowiedzi gracza. W tym przypadku użytkownik wybiera z listy gotową odpowiedź, zamiast jedynie tonu jak w grze *Mass Effect*. Pozwala to na większą kontrolę przez gracza oraz umożliwia uniknięcie sytuacji, w której gracz spodziewał się innej odpowiedzi, wybierając daną opcję dialogową.



Rysunek 2.6: Przykład koła dialogowego w grze *Mass Effect*.



Rysunek 2.7: Kadr z gry *Fallout 3* przedstawiający przykładowy dialog.

## 2.7. Sterowanie jednostkami w grach (Zofia Sosińska)

Gry z możliwością tworzenia drużyny muszą rozwiązać problem zachowania podwładnych. Program może udostępniać skomplikowaną sztuczną inteligencję dla wojowników, zawsze rozwiązującą konflikt w optymalny sposób. W tym przypadku użytkownik staje się obserwatorem, a nie dowodzącym, co go znacznie odciąży. Innym podejściem może być zaprojektowanie podwładnych jako marionetek bez własnej woli, które będą biernie czekać, aż do otrzymania rozkazu. Wtedy jednak gracz musi skupić się na każdym ruchu zarówno swojej, jak i przeciwniej drużyny tak, aby w porę móc zareagować na wszystkie zmiany, co jest rozwiązaniem obciążającym dla użytkownika. Przy projektowaniu mechaniki sterowania jednostkami trzeba zachować balans pomiędzy dodaniem i odjęciem użytkownikowi zadań, na których musi się skupić. Dla każdej gry ta proporcja może być inna, zależnie od unikalnego charakteru gry. "Gry należą do kategorii [...], w której trudno jest ocenić, czy zachowany jest balans. Nie możemy ich włożyć do «maszyny ważcej», która zmierzyłaby zbalansowanie." [14].

Przy projektowaniu gry *Mount&Blade*<sup>17</sup> studio TaleWorlds Entertainment zdecydowało się zaimplementować mechanikę sterowania jednostkami tak, aby dodać do walki element strategii. Gracz bezpośrednio steruje jedynie główną postacią i podczas walki może wydawać rozkazy reszcie swojej drużyny. Poprzez cyfry 0-4 wybiera grupę jednostek, do której się odnosi np. łuczników. Po naciśnięciu konkretnego przycisku, po lewej stronie ekranu pojawia się lista dostępnych komend. Następnie przez klawisze F1-F11 wydaje konkretny rozkaz np. odwrót. Lista znika, a sztuczna inteligencja postaci zajmuje się już samym wykonaniem czynności. Gracz nie martwi się, czy jednostki znajdą optymalną drogę, będą celować w przeciwników, czy z nimi walczyć. Zachowany jest więc przyjemny balans pomiędzy podejmowaniem kluczowych decyzji, a byciem odciążonym przez mechanizmy sztucznej inteligencji.

<sup>17</sup><https://www.taleworlds.com/en/Games/MountAndBlade>



Rysunek 2.8: Wykaz dostępnych rozkazów z gry *Mount&Blade*.

### **2.8. Kompasy w grach (Zofia Sosińska)**

Częstą funkcjonalnością gier jest możliwość eksploracji świata, w którym na gracza czekają przygody i zadania do wykonania. Jednak przestrzeń przygotowana dla użytkownika może być na tyle duża i skomplikowana, że powstaje ryzyko zgubienia się. Jednym ze sposobów jakim projektanci gier wyciągają rękę do użytkownika jest danie mu kompasu. Może to być bardzo proste narzędzie, pokazujące jedynie strony świata, immitujące klasyczny przedmiot ze świata rzeczywistego. Już taka garstka informacji potrafi uproszczyć graczyowi odnalezienie drogi do celu. Jednak innym podejściem jest rozszerzenie funkcjonalności kompasu o pokazywanie pozycji innych ważnych punktów odniesienia, celów, do których trzeba się dostać lub kluczowych postaci.

*The Elder Scrolls V: Skyrim* (skrótnie *Skyrim*)<sup>18</sup> jest to fabularna gra wyprodukowana przez Bethesda Game Studios i wydana przez Bethesda Softworks. Autorzy przygotowali dla gracza duży świat, który ten może dowolnie eksplorować. Nawigację oparli o bardzo pomocne i sprytne rozwiązanie, jakim jest pasek przedstawiający pole widzenia gracza. Służy on między innymi jako kompas, ponieważ jedną z jego mechanik jest pokazanie użytkownikowi stron świata, znajdujących się w kierunku, w którym on patrzy. Pasek ułatwia także poruszanie się po świecie, sygnalizując położenie wrogów, kompanów i ważnych dla rozgrywki lokalizacji.



Rysunek 2.9: Kompas z gry *Skyrim*.

### **2.9. Sterowanie postacią oraz walka (Bogna Lew)**

Udostępnianie graczyowi jego własnej postaci jest cechą charakterystyczną raczej komputerowych gier fabularnych niż gier strategii czasu rzeczywistego. Jednakże jest to ciekawe rozwiązanie, które w znaczny sposób urozmaica rozgrywkę oraz wpływa na stopień zaangażowania użytkownika. Udostępnienie graczyowi jego własnej postaci "czyni gracza kreatywnym elementem działającym wewnątrz dyskursu, który posiada przestrzenny charakter." [8].

Przykładem gry implementującej łatwy w obsłudze sposób sterowania postacią jest gra *The Elder*

<sup>18</sup><https://elderscrolls.bethesda.net/en>

*Scrolls V: Skyrim*. Umożliwia ona graczowi możliwość poruszania się postacią za pomocą klawiszy W, A, S oraz D. Dodatkowo użytkownik może zmieniać prędkość swojego bohatera, przytrzymując klawisze Alt bądź Ctrl, które odpowiednio powodują zwolnienie lub przyspieszenie tempa przemieszczania się. Do rozglądzania się wykorzystywana jest mysz, której ruch powoduje obrót postaci wokół własnej osi. Ataki gracz może wykonać poprzez naciśnięcie prawego bądź lewego przycisku myszy, które powodują wykonanie akcji odpowiednio lewą lub prawą dlonią. Jest to przykład dbałości o szczegóły, które sprawiają, że gra jest jeszcze bardziej interesująca i satysfakcyjną.

Innym tytułem wartym uwagi jest *Kingdom Come: Deliverance*<sup>19</sup>. Jest to gra osadzona w realiach Europy Środkowej na początku XV wieku. Godny uwagi jest jej mechanizm walki, ponieważ twórcy skupili się na jak najdokładniejszym oddaniu średniowiecznego stylu walki. W tym celu skrupulatnie przestudiowali, w jaki sposób władało mieczem w tamtych czasach, a następnie w pełni oddali to w grze. Wykorzystali do tego tysiące animacji oraz starannie oddali fizykę pojedynków. W efekcie powstał realistyczny mechanizm walki, który umożliwia graczowi parowanie, zadawanie ciosów oraz blokowanie.

Kolejnym interesującym tytułem jest gra *Mount&Blade*. Podobnie jak w przypadku *Skyrima*, gracz steruje swoją postacią za pomocą klawiszy W, A, S oraz D. Co więcej, gracz może wykonywać ataki poprzez naciśnięcie lewego przycisku myszy, co jest typowym rozwiązaniem w grach, które w prosty sposób umożliwia graczowi uczestniczenie w potyczkach. Jednakże w przeciwieństwie do *Skyrima*, w *Mount&Blade* ruch myszą nie powoduje obrotu całej postaci, a jedynie kamery. Zmiana kierunku odbywa się wyłącznie za pomocą klawiszy sterujących. Dzięki temu gracz może zobaczyć, co się dzieje za jego postacią bez konieczności zmiany kierunku ruchu bądź zatrzymania się. Powoduje to jednak pewne kłopoty z zachowaniem realizmu, gdyż w prawdziwym świecie nie jest możliwe zobaczenie czegoś bez konieczności zwrócenia się w tę stronę.

Powyższe przykłady obrazują najpopularniejsze rozwiązania umożliwiające sterowanie postacią oraz wykonywanie ataków. Typowymi narzędziami wykorzystywanymi do poruszania postacią oraz kamerą są myszka i klawisze W, A, S oraz D, natomiast do atakowania - lewy przycisk myszy. Różnice zaczynają się pojawiać w samych mechanikach, gdyż to w jaki sposób postać gracza będzie się zachowywać zależy od wizji autorów gry.

## 2.10. Rozwój postaci gracza w grach (*Bartosz Strzelecki, Bogna Lew*)

Rozwój postaci gracza ma ogromne znaczenie w wielu wymiarach. Jest podstawowym elementem, dzięki któremu gracz może odczuwać postęp podczas rozwoju linii fabularnych. Stanowi to też nagrodę dla gracza, za poniesioną głęboką osobistą inwestycję wczuwając się w rolę postaci. Przez to ta mechanika pozytywnie wpływa na motywację użytkownika do dalszych zmagań i pozwala na ponowne rozegranie gry, dzięki temu, że gracz może chcieć przejść ponownie przez linię fabularną, podejmując inne decyzje podczas rozwijania postaci.

Jednym z wyróżniających aspektów serii gier *The Elder Scrolls* jest zaimplementowany system rozwoju postaci, który pozwala na różnorodne przebiegi rozgrywki dynamicznie dostosowujące się do stylu preferowanego przez gracza. W skrócie, jeśli gracz preferuje strzelanie z łuku, w trakcie rozgrywki będzie mógł rozwijać tę umiejętność, aby czerpać z niej jak najwięcej korzyści. Głęboka złożoność tego systemu sprzyja długoterminowemu zaangażowaniu, dzięki czemu gracze poświęcają dużo więcej czasu na doskonalenie swoich postaci, odkrywanie nowych kombinacji umiejętności i eksperymentowaniem z różnymi stylami rozgrywki.

---

<sup>19</sup><https://www.kingdomcomerpg.com/pl>



Rysunek 2.10: Ekran rozwoju postaci z gry *TESV: Skyrim*.

Innym przykładem jest rozwój postaci w grze *Wiedźmin 3: Dziki Gon*, w której gracz wciela się w rolę wiedźmina Geralta. Gra udostępnia cztery kategorie umiejętności bohatera, które użytkownik może rozwijać poprzez przyznawanie punktów. Gra przyznaje je graczowi po tym jak zdobycie on odpowiedni poziom doświadczenia, który może podnosić poprzez wykonywanie zadań, czy zabijanie potworów. Każda z umiejętności, które gracz zdecyduje się rozwijać wymaga wcześniejszej aktywacji, aby przynosić mu korzyści w trakcie rozgrywki. Jest to ciekawy mechanizm, który wymusza na graczu przemyślenie w jaki sposób chce kształtować swoją postać i obmyślenie planu jej rozwoju.



Rysunek 2.11: Ekran rozwoju postaci z gry *Wiedźmin 3: Dziki Gon*.

## **2.11. Przekazywanie informacji o świecie w grach (Bartosz Strzelecki)**

Systemy przekazywania graczowi informacji o świecie są nieocenioną pomocą w kształtowaniu dynamiki i dostarczaniu kluczowych informacji. Jest to podstawowy element rozgrywki, pozwalający na dynamiczne podejmowanie decyzji oraz na koordynację działań w grach wieloosobowych.

Gra *Dead by Daylight*<sup>20</sup> jest asymetryczną grą wieloosobową, w której gracze wcielają się w role postaci "ocalałych" starających się uciec z mapy albo w rolę "zabójcy", którego celem jest niedopuszczenie do tego. Jednym z udostępnianych przez tę grę mechanizmów jest mechanika widzenia przez przeszkody, która stanowi istotny element rozgrywki, zapewniający dodatkową warstwę strategii. Polega na wyświetlaniu reprezentacji odległych celów, przedmiotów i przeciwników zakrytych przez przeszkody. Ta zdolność odgrywa kluczową rolę dla obu stron konfliktu.

W przypadku "ocalałych", ta mechanika jest dostępna dzięki atutom i przedmiotom. Mechanika ujawnia lokalizację celów oraz innych "ocalałych" pozwalając na koordynację i opracowanie strategii wspólnych działań.

I odwrotnie, zdolność "zabójcy" do widzenia aury jest kluczowa dla jego mechaniki rozgrywki. Aury umożliwiają im śledzenie "ocalałych", zwłaszcza gdy korzystają z ich unikalnych mocy lub specyficznych atutów. Mechanika ta zwiększa napięcie, ponieważ "ocalali" muszą zachować czujność i strategiczne podejście, aby uniknąć pola widzenia "zabójcy" lub zakłócić ich zdolność czytania aury, aby uciec i osiągnąć cele.

Ogólnie rzecz biorąc, mechanika aury w Dead by Daylight służy jako podstawowy element rozgrywki, który równoważy wymianę informacji między "ocalałymi" a "zabójcami", znacząco przyczyniając się do atmosfery napiecia i strategii w grze.

<sup>20</sup><https://deadbydaylight.com>



Rysunek 2.12: Przykładowe elementy widzenia przez horyzont w grze *Dead by Daylight*.



Rysunek 2.13: Przykładowy efekt aury w grze *Dead by Daylight*.

## 2.12. Modele celowania w grach (Bartosz Strzelecki)

Celowanie w grach taktycznych jest podstawowym elementem programu, który ma olbrzymi wpływ na przebieg rozgrywki. Ten mechanizm jest przede wszystkim wykorzystywany do wzbogacenia gry o kolejną warstwę decyzji taktycznych i zarządzania podejmowanym ryzykiem. Gracz, przed oddaniem strzału, podejmuje decyzję czy warto zaryzykować podjęcie strzału o niskiej szansie na trafienie, czy lepiej wykonać niezawodny atak, prawdopodobnie wystawiając się na ataki przeciwników.

W *Phoenix Point*<sup>21</sup> modelowanie celności to wieloaspektowy system, który w zawiły sposób definiuje wynik interakcji bojowych. Gra wykorzystuje dynamiczny system celowania, który uwzględnia różne elementy, takie jak postawa żołnierza, biegłość w posługiwaniu się bronią, zasięg, osłona i warunki środowiskowe, aby określić precyzję strzału. Każdy z tych elementów odgrywa znaczącą rolę w ogólnym obliczeniu trafienia w cel.

W przeciwieństwie do podobnej gry *XCOM*<sup>22</sup>, gdzie celność jest zamodelowana za pomocą prostej szansy na trafienie, w grze *Phoenix Point* trajektoria każdego pocisku obliczana jest osobno. Podczas celowania widoczne są dwa okręgi: wewnętrzny, który reprezentuje miejsce, w którym znajdzie się 50% pocisków oraz zewnętrzny, który reprezentuje maksymalny rozrzut broni. W tym przypadku im celniejsza broń tym okręgi będą mniejsze.



Rysunek 2.14: System celowania występujący w grze *Phoenix Point*.

Ostatecznie system zaimplementowany w grze *Phoenix Point* okazuje się dużo bardziej realistyczny i pozwala na utrzymywanie ciągłej niepewności co do celności strzału, umożliwiając w ten sposób na strategiczne wyzwania, leżących u podstaw gier tego typu.

<sup>21</sup><https://phoenixpoint.info>

<sup>22</sup><https://www.xcom.com>

### **3. TECHNOLOGIE, ALGORYTMY I NARZĘDZIA**

W niniejszym rozdziale zostały omówione wykorzystywane w trakcie pracy narzędzia wraz z przeglądem wybranych silników gier komputerowych. Dodatkowo przedstawiono wykorzystane w projekcie biblioteki i rozwiązań służące do zarządzania strukturami danych.

#### **3.1. Przegląd silników gier (Bogna Lew)**

Silnik gier komputerowych jest oprogramowaniem służącym głównie do wytwarzania gier komputerowych. "Silnik zawiera narzędzia i komponenty, które obsługują różne podstawowe elementy gry, takie jak rysowanie grafiki, kontrolowanie dźwięku i przesuwanie obiektów, i pozwalają programistom skupić się na generowaniu danych dodatkowych do silnika, takich jak modele/obrazy, tekst, efekty dźwiękowe i tak dalej" [15]. Wybranie odpowiedniego silnika przed rozpoczęciem pracy ma kluczowy wpływ na proces twórczy i jego efekt.

Obecnie dostępnych jest wiele silników, a każdy z nich ma różne możliwości. W celu uproszczenia wyboru zdecydowaliśmy się zawieźć listę do trzech najpopularniejszych obecnie darmowych silników, czyli Godot, Unity oraz Unreal Engine.

Pierwszy z nich jest w pełni darmowym silnikiem open source. Posiada prosty i intuicyjny interfejs, a w Internecie tworzone jest przez społeczność wiele samouczków. Nie posiada on jednak oficjalnej dokumentacji oraz jest zdecydowanie mniej popularny od pozostałych dwóch.

Kolejny silnik, Unity jest określany jako przyjazny dla początkujących. Posiada bogatą dokumentację oraz jest dostępnych dużo samouczków stworzonych przez jego społeczność. Unity świetnie się nadaje do tworzenia gier 3D. Silnik ten jest dostępny w wersji bezpłatnej oraz oferującej więcej możliwości wersji płatnej. Co więcej, ma możliwość rozszerzenia o dodatkowe narzędzia dostępne w Asset Store.

Ostatni z silników jest najbardziej kojarzony z grami AAA. Cechuje go zaawansowana grafika, która umożliwia wytwarzanie fotorealistycznych gier. Korzystanie z niego jest darmowe, a opłata w wysokości 5% jest naliczana jedynie, gdy gra zarobi ponad milion USD.

Tabela 3.1 przedstawia porównanie wymienionych silników w istotnych, z punktu widzenia projektu, aspektach.

**Tabela 3.1:** Porównanie silników.

Silnik	Unity	Unreal Engine	Godot
Popularność	duża	duża	mała
3D	Tak	Tak	Tak
Język	C#	C++	C#, C++, GDScript
Baza wiedzy	dokumentacja, samouczki	dokumentacja, samouczki	samouczki, fora
Open source	Nie	Nie	Tak

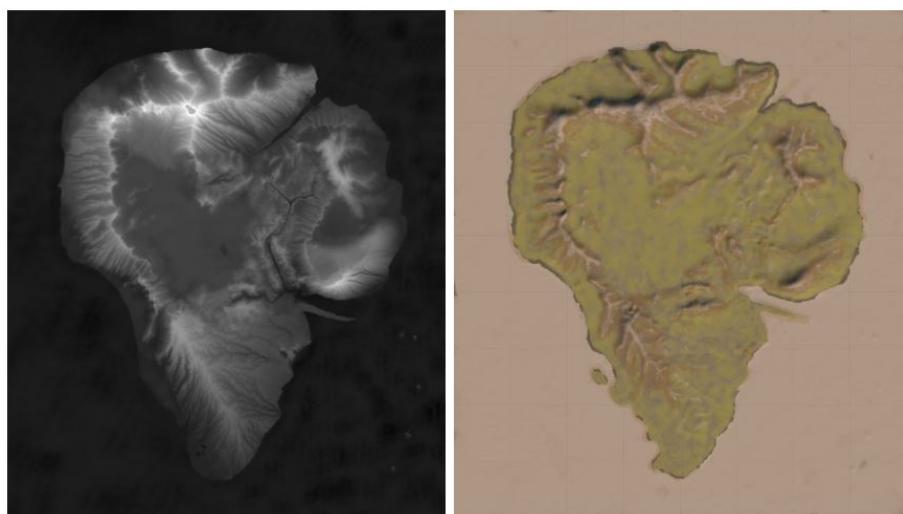
Finalnie zdecydowaliśmy się na implementację gry w Unity, ponieważ jest to silnik, który najlepiej odpowiada wymaganiom projektu.

## 3.2. Narzędzia dostępne dla silnika Unity

### 3.2.1. Terrain Toolbox (Bogna Lew)

Terrain Toolbox jest narzędziem dostępnym dla silnika Unity. Jest to zasób dostępny w paczce Terrain Tools, który upraszcza pracę nad modelowaniem terenu do gry.

Do podstawowych funkcjonalności udostępnianych przez Terrain Toolbox należy generowanie terenu na podstawie map wysokości (ang. *heightmap*) oraz podstawowych parametrów takich jak długość, szerokość i wysokość terenu. Pozwala to na szybkie utworzenie grywalnej mapy. Ponadto narzędzie Terrain Toolbox umożliwia wygładzenie, dodatkowe wymodelowanie oraz nałożenie tekstur na tak utworzony teren za pomocą udostępnionych przez nie funkcjonalności.



Rysunek 3.1: Przykładowa mapa wysokości (po lewej) oraz wygenerowany na jej podstawie teren (po prawej).

Kolejną istotną funkcjonalnością jest malowanie terenu drzewami. Umożliwia ona automatyczne umiejscowienie obiektów na mapie w losowy sposób. Pozwala to szybko utworzyć realistyczne skupiska obiektów, takich jak las. Poniżej został przedstawiony przykładowy rezultat.



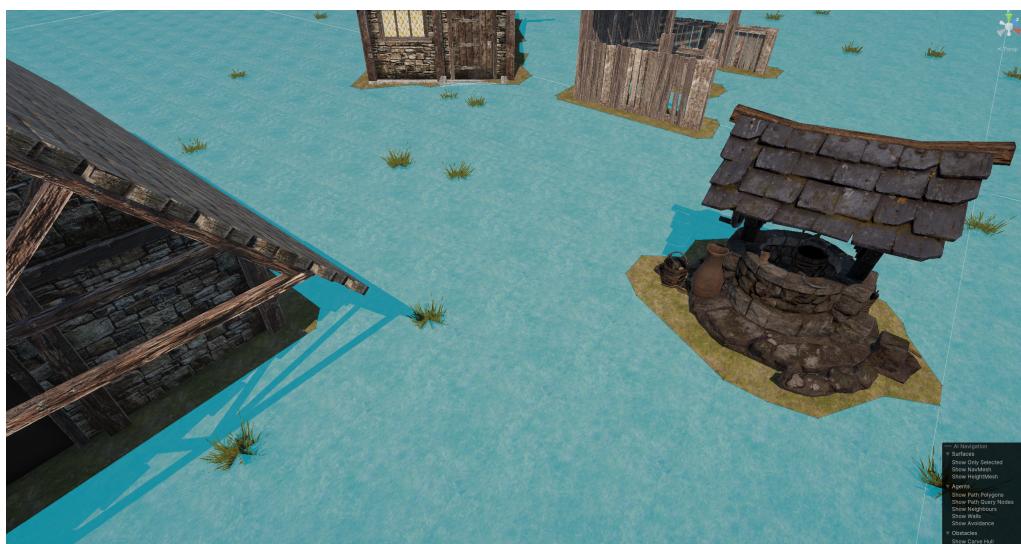
Rysunek 3.2: Widok na teren z drzewami.

### 3.2.2. *Navmesh (Bartosz Strzelecki)*

Komponent Unity Navmesh jest podstawowym elementem wykorzystywanym w przypadku znajdywania ścieżek. Jest to mechanizm pozwalający na przechowywanie wyspecjalizowanych danych, które reprezentują powierzchnie, po których mogą poruszać się agenci sztucznej inteligencji. W skrócie Na-vMesh to "Siatka generowana przez Unity w celu przybliżenia obszarów, po których można chodzić i przeszkód w otoczeniu, w celu znalezienia ścieżki i nawigacji kontrolowanej przez sztuczną inteligencję" [16].

Aby zacząć korzystać z komponentu należy wygenerować mapę w edytorze Unity. W ramach tego procesu wykonywane są obliczenia mające na celu wykrycie powierzchni, po której może poruszać się dany agent. Ten system bierze pod uwagę kąt nachylenia powierzchni, szerokość przejścia, jak i wysokość stropu.

W trakcie rozgrywki komponent NavMesh Agent wykorzystuje wcześniej wygenerowane dane do wyznaczenia najlepszej ścieżki do celu. Z poziomu skryptów można dynamicznie modyfikować powierzchnię nawigacyjną, umożliwiając w ten sposób uzyskanie poruszających się przeskódeł i dynamicznie powstających budynków.



Rysunek 3.3: System nawigacji w Unity.

Podsumowując, system NavMesh istotnie upraszcza zadanie odnajdywania ścieżki, co pozwala na łatwe zaimplementowanie realistycznych zachowań agentów sztucznej inteligencji. Znacznie ułatwia też zadanie tworzenia świata gry ze względu na automatyczny sposób generowania danych nawigacyjnych.

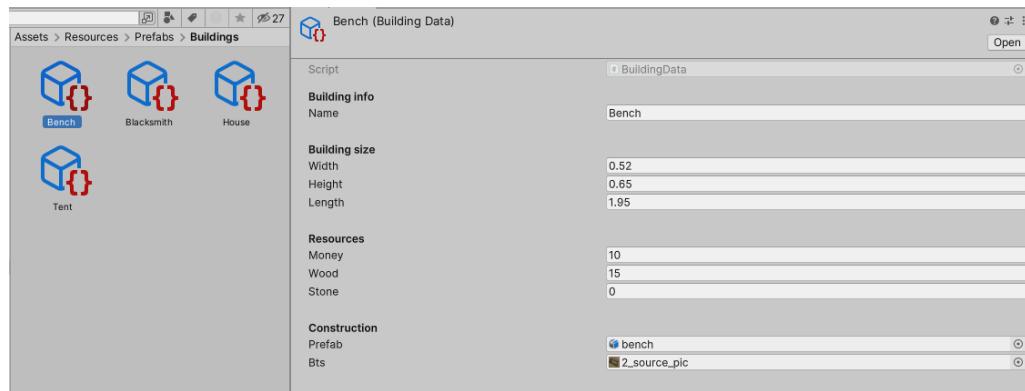
## 3.3. *Struktury danych*

### 3.3.1. *ScriptableObject (Bogna Lew)*

Silnik Unity umożliwia użytkownikom tworzenie klas typu ScriptableObject. Jest to struktura danych, która pozwala na tworzenie wielu instancji klasy bez konieczności kopowania danych. Poszczególne instancje współdzielą informacje, dzięki czemu możliwe jest znaczące zoptymalizowanie użycia pamięci.

Do podstawowych zalet ScriptableObject należy możliwość wykorzystania go do tworzenia zasobów, które można by wykorzystać w trakcie gry. Dzięki temu w prosty sposób można utworzyć szablony dla

obiektów takich jak budowle, bronie i inne przedmioty, które następnie można użyć do utworzenia ich instancji podczas rozgrywki.



Rysunek 3.4: Przykład zasobów typu ScriptableObject.

### 3.3.2. *Protobuf* (Bartosz Strzelecki)

Biblioteka Protobuf jest wydajnym i uniwersalnym rozwiązaniem przystosowanym do serializacji danych. Służy do zdefiniowania formatu przechowywanych informacji, który jest neutralny dla platformy. "Protocol Buffers zapewniają format serializacji dla pakietów o typowanych i ustrukturyzowanych danych o rozmiarze do kilku megabajtów. Format nadaje się zarówno do efemerycznego ruchu sieciowego, jak i długoterminowego przechowywania danych. Protocol Buffers można rozszerzyć o nowe informacje bez pozbawienia ważności istniejących danych lub konieczności aktualizacji kodu." [17]. W swojej istocie Protobuf definiuje niezależny od języka programowania schemat opisu interfejsu, który pozwala na określenie struktury danych za pomocą prostej i intuicyjnej składni. Elastyczność komponentu Protobuf wykracza poza proste struktury danych, oferując obsługę złożonych typów danych, pól opcjonalnych i powtarzalnych, a także struktur zagnieżdzonych. Dodatkowo udostępnia narzędzia do generowania kodu, które automatycznie tworzą implementację specyficzną dla danego języka, ułatwiając pracę programistom.

## 4. PROJEKT SYSTEMU

W tym rozdziale został przedstawiony zamysł członków zespołu na opracowywaną grę. Stanowi on opis pożądanych rezultatów, do których zespół będzie dążyć w trakcie implementacji. Kolejno zostały opisane oczekiwane działanie poszczególnych mechanik, z których będzie się składał ostateczny produkt.

### 4.1. Organizacja (*Bartosz Strzelecki*)

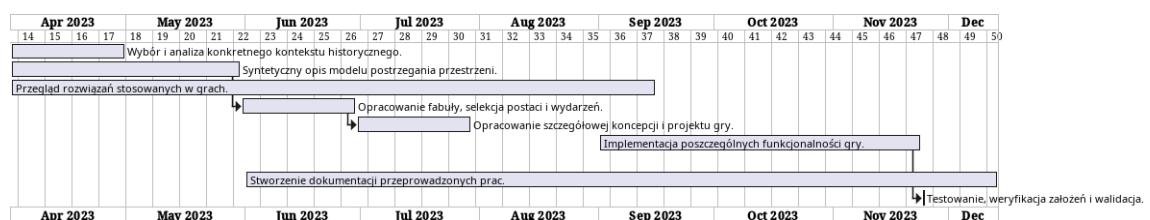
W tym podrozdziale przedstawiono harmonogram prac, wraz z ich przewidywanym terminem realizacji. Ponadto zaprezentowano skład zespołu projektowego, kompetencje ich członków oraz podział zadań.

#### 4.1.1. Główne etapy projektu

Poniższa tabela przedstawia harmonogram prac w postaci kamieni milowych. Opisuje szacowany termin zakończenia danych etapów.

Etap	Termin realizacji
Wybór i analiza konkretnego kontekstu historycznego.	Kwiecień 2023
Syntetyczny opis modelu postrzegania przestrzeni na podstawie dzieł pisanych, architektury i sztuki.	Kwiecień — Maj 2023
Przegląd rozwiązań stosowanych w grach strategicznych z wybranego okresu oraz dodatkowo mechanizmów z innych gier, które mogłyby być zaadoptowane na potrzeby projektu.	2, 3 kwartał 2023
Opracowanie fabuły, selekcja postaci i wydarzeń, a także określenie zakresu autonomii świata gry oraz możliwości modyfikowania go przez gracza.	Czerwiec 2023
Opracowanie szczegółowej koncepcji i projektu gry, w tym projekt mechanizmów zawartych w prototypie.	Lipiec 2023
Implementacja poszczególnych funkcjonalności gry.	4 kwartał 2023
Testowanie, weryfikacja założeń i walidacja.	Listopad — Grudzień 2023
Stworzenie dokumentacji przeprowadzonych prac.	3, 4 kwartał 2023

Przewidywany termin zakończenia prac nad projektem to grudzień 2023 roku.



Rysunek 4.1: Harmonogram przedstawiony w postaci diagramu gantt.

## **4.2. Skład zespołu projektowego**

W niniejszej sekcji przedstawiono skład zespołu realizującego projekt dyplomowy oraz główne obszary wykonywanych prac.

Imię i nazwisko	Bogna Lew	Zofia Sosińska	Bartosz Strzelecki
Zadania	System budowania, sterowanie postacią	Interfejs użytkownika	Sztuczna inteligencja postaci

## **4.3. Analiza i specyfikacja wymagań (Bogna Lew, Zofia Sosińska)**

W niniejszej sekcji przedstawiono specyfikę wymagań funkcjonalnych, pozafunkcjonalnych oraz tych, wynikających z głównych założeń projektu. Dodatkowo zawiera ona diagramy przypadków użycia, maszyny stanów oraz klas prototypowej gry.

### *4.3.1. Specyfika wymagań wynikających z założeń projektu*

Z punktu widzenia projektu kluczowe jest jak najdokładniejsze oddanie realiów historycznych przy jednoczesnym uwzględnieniu jakości rozgrywki gracza oraz cech charakterystycznych dla gier typu RTS. Z założeń wynika, że fabuła gry powinna zostać osadzona w czasach sprzed wielkich odkryć geograficznych. Na tej podstawie zostały zdefiniowane dodatkowe wymagania, które powinien spełniać prototyp.

Gra powinna zawierać:

- sposób nawigacji jak najdokładniej odpowiadający temu stosowanemu w wybranej epoce,
- postacie:
  - stylistycznie pasujące do realiów historycznych,
  - wykorzystujące słownictwo adekwatne do czasów, w których osadzona jest gra,
  - jak najlepiej oddające światopogląd w danych czasach,
  - stosujące orędzia typowe dla wybranej epoki,
- budowle stylistycznie odpowiadające wybranej epoce,
- sposób komunikacji z postaciami imitujący ten stosowany w danych czasach.

Dodatkowo po konsultacjach z pomysłodawcą projektu zdecydowano, że prototyp gry nie ma być typową grą z gatunku strategii czasu rzeczywistego, a jego hybrydą z gatunkiem komputerowych gier fabularnych.

### *4.3.2. Wymagania funkcjonalne*

Niniejsza sekcja skupia się na określeniu wymagań funkcjonalnych, które powinien spełniać prototyp gry.

Gra powinna oferować możliwość:

- uruchomienia nowej gry,
- sterowania postacią gracza,
- nawigacji w świecie gry,
- wchodzenia w interakcję z postaciami niezależnymi,
- przyjmowania zleceń od postaci niezależnych,

- najmowania postaci wojowników,
- wydawania komend wynajętym postaciom,
- zlecania budowy,
- zdobywania zasobów,
- odczytu wybranego stanu gry z komputera użytkownika,
- zapisu aktualnego stanu gry lokalnie na komputerze użytkownika.

#### 4.3.3. Wymagania pozafunkcjonalne

W tej sekcji zostały przedstawione wymagania pozafunkcjonalne projektu.

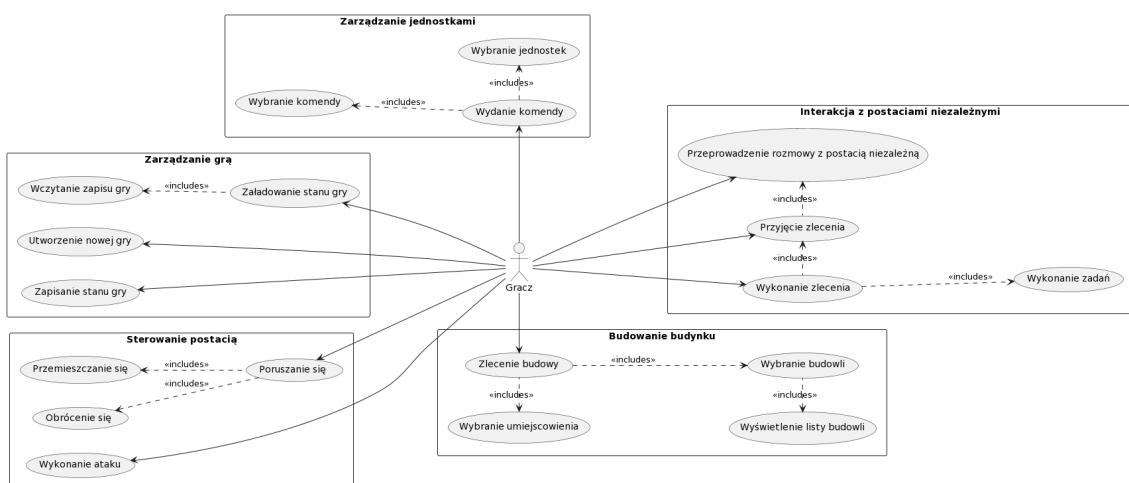
Gra powinna umożliwiać:

- rozgrywkę w trybie offline,
- działanie na urządzeniach z systemem Windows lub Linux,
- dostosowywanie rozmiaru do wielkości ekranu komputera użytkownika,
- obsługę klawiatury oraz myszy,
- działanie w czasie rzeczywistym.

#### 4.3.4. Diagram przypadków użycia

Niniejsza sekcja przedstawia diagram przypadków użycia dla głównych funkcjonalności, które będzie zawierać prototypowa gra. Opisuje on przewidywane usługi oferowane przez poszczególne mechaniki programu.

Jedną z głównych akcji, które gra udostępni będzie wydanie rozkazów przyjaznym jednostkom. Polegać będzie ona na poinformowaniu wojowników przez gracza jaką czynność powinni w danym momencie wykonać. Kolejną możliwością będzie zlecenie budowy, czyli zlecenie budowniczemu wybudowania wybranego obiektu w określonym przez gracza miejscu. Ponadto użytkownik będzie mógł przeprowadzać rozmowy z postaciami niezależnymi. Oznacza to, że będzie mógł zainicjować z nimi dialog i następnie kształtać jego przebieg poprzez wybieranie swojej odpowiedzi z opcji proponowanych przez grę.



Rysunek 4.2: Diagram przypadków głównych mechanik gry.

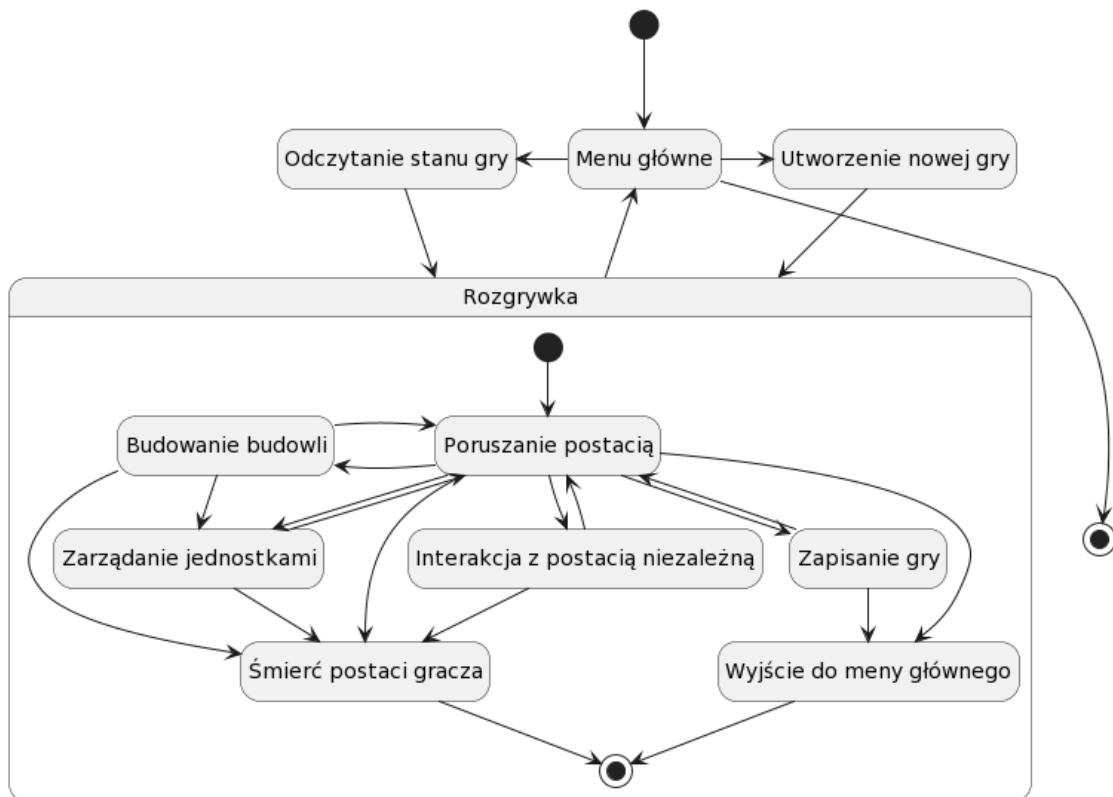
#### 4.3.5. Diagram stanów

W tym podpunkcie został przedstawiony diagram stanów prototypowej gry, który ukazuje jej przewidywany sposób działania. Prezentuje on podstawowe stany, w których może się znaleźć system gry.

Do podstawowych stanów należą "Menu główne" oraz "Rozgrywka". Pierwszy z nich oznacza, że program został uruchomiony, a gracz wyświetla panel główny. Z tego stanu możliwe jest przejście do stanów "Odczytanie stanu gry" bądź "Utworzenie nowej gry", które to powodują rozpoczęcie gry z zapisu lub od początku.

Stan "Rozgrywka" jest stanem złożonym i określa, że gra została rozpoczęta. W jego skład wchodzą przede wszystkim takie stany, jak "Interakcja z postacią niezależną", w którym program się znajdzie, gdy gracz rozpocznie dialog z postacią w grze, czy "Zarządzanie jednostkami", który to oznacza, że użytkownik wydaje rozkazy swoim wojownikom.

Diagram stanów został przedstawiony na rysunku 4.3.



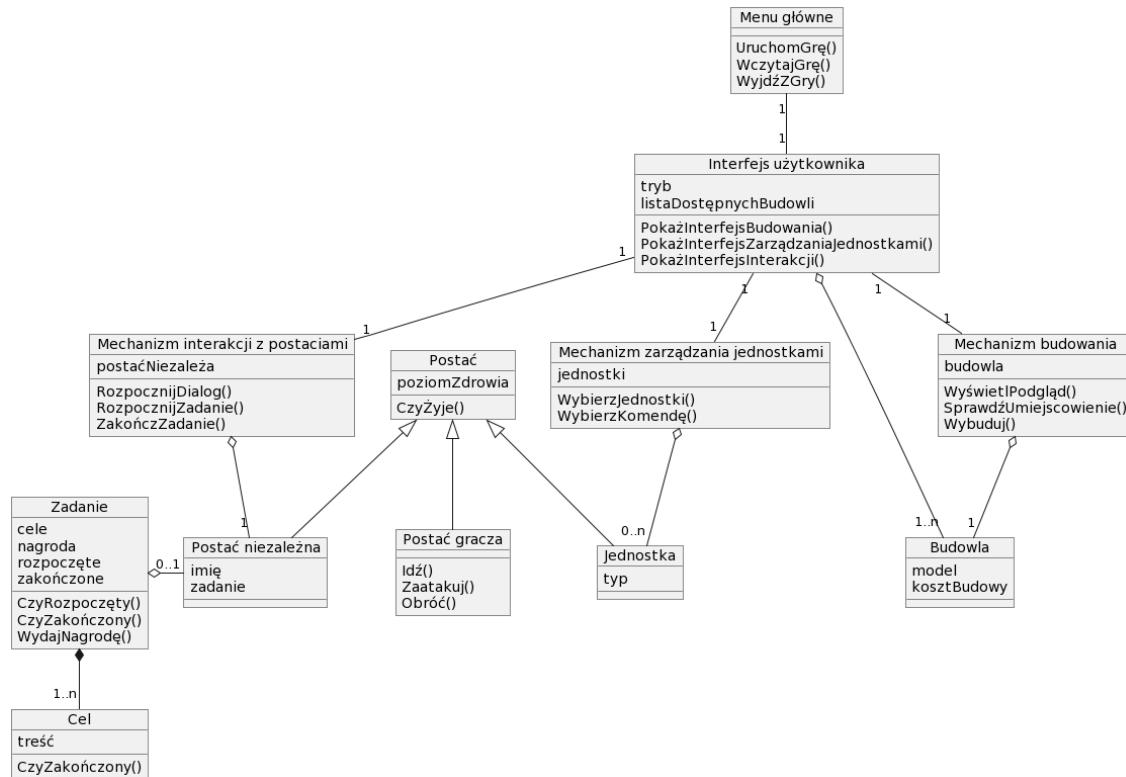
Rysunek 4.3: Diagram stanów gry.

#### 4.3.6. Diagram klas

W tej sekcji został pokazany uproszczony diagram klas (rys. 4.4), przedstawiający główne elementy gry. Obrazuje podstawową strukturę tworzonego systemu oraz zależności pomiędzy poszczególnymi komponentami.

Do najważniejszych klas należą "Interfejs użytkownika", "Mechanizm interakcji z postaciami", "Mechanizm zarządzania jednostkami" oraz "Mechanizm budowania". Obrazują one podstawowe komponenty gry, których głównym zadaniem jest zarządzanie poszczególnymi mechanikami. "Interfejs użytkownika" jest odpowiedzialny za interakcję z graczem oraz pomaganie mu w trakcie rozgrywki. Pozostałe trzy kolejno pozwalają graczowi na prowadzenie dialogów z postaciami niezależnymi, wydawanie

komend jego zaprzyjaźnionym jednostkom oraz budowanie obiektów.



Rysunek 4.4: Diagram klas gry.

#### **4.4. Opis świata gry i charakteru rozgrywki (Bogna Lew)**

Fabuła wytwarzanej gry ma zostać osadzona w realiach wczesnego średniowiecza i została ona zainspirowana kulturą celtycką. W tym okresie można było ich spotkać głównie w Irlandii, więc z tego powodu mapa świata gry będzie prezentować górzystą wyspę, na której gracz będzie mógł znaleźć niewielką wioskę oraz obozowiska.

Graczowi udostępniona zostanie jego własna postać do bezpośredniego sterowania. Elementem gry będą zadania poboczne, które będą stanowić urozmaicenie rozgrywki i dodatkowo zachętą gracza do zagłębiania się w nią. Z tego powodu w trakcie gry użytkownik będzie mógł spotkać postaci niezależne, z którymi możliwe będzie wejście w interakcje, kończące się np. zleceniem wykonania zadania. W ich realizacji będą mu pomagać jednostki, z którymi się zaprzyjaźni podczas rozgrywki i którym będzie mógł wydawać komendy zgodnie z ich typem. Dodatkowo w trakcie eksploracji świata natrafi na nieprzyjazne postacie, z którymi będzie toczyć walki. Grę urozmaicą postaci zwierząt, które mogą być neutralne, bądź agresywne wobec gracza.

Gra będzie udostępniać trzy podstawowe surowce, za które gracz będzie mógł budować budynki. Przewidywane są dwa sposoby ich pozyskiwania. Pierwszym z nich jest wykonywanie zadań, za które może uzyskać nagrody w postaci pewnej ilości surowców. Kolejnym sposobem jest zbieranie kłód drewna oraz kamieni leżących na ziemi. Podnoszenie ich dostarczy jednorazowy przypływ odpowiadającego im zasobu.

#### **4.5. Interfejs użytkownika (Zofia Sosińska)**

Projekt interfejsu użytkownika przewiduje tryb podstawowy z zawsze widocznymi elementami oraz dynamicznie pojawiające się okna, wywoływane za pomocą konkretnych klawiszy. Zadaniem każdej składowej będzie odzwierciedlenie pewnego fragmentu aktualnego stanu wiedzy granej postaci z naciskiem na najpotrzebniejsze w danej chwili informacje. Przewidziane są:

- menu stawiania budynków,
- menu wydawania komend,
- menu zapisu,
- informacja o możliwej interakcji,
- dziennik z zadaniami,
- ekran końca gry.

##### *4.5.1. Interfejs podstawowy*

Interfejs podstawowy przewiduje funkcje, takie jak pokazanie:

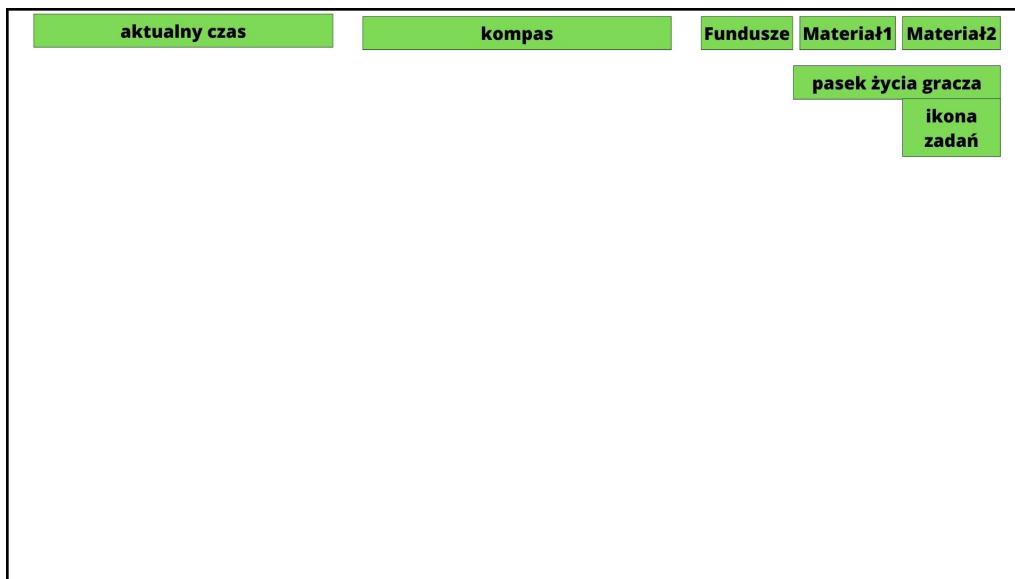
- aktualnego czasu w grze, aby stworzyć iluzję upływającego czasu w świecie gry,
- surowców i funduszy, aby użytkownik nie był obarczony kalkulacjami przy każdym zakupie lub przypływie zasobów,
- kompasu, aby ułatwić nawigację,
- stan zdrowia gracza, aby zasygnalizować mu, czy przypadkiem nie rozsądniejsze będzie wycofanie się z potyczki,
- etap, na którym są przypisane graczu zadania, aby przypomnieć mu, że na takowe się zgodził.

Inspiracją dla górnego paska z informacjami jest ten użyty w grze *Warcraft 3* (por. 2.5). Prostota i surowość stylu będą współgrać z klimatem gry.

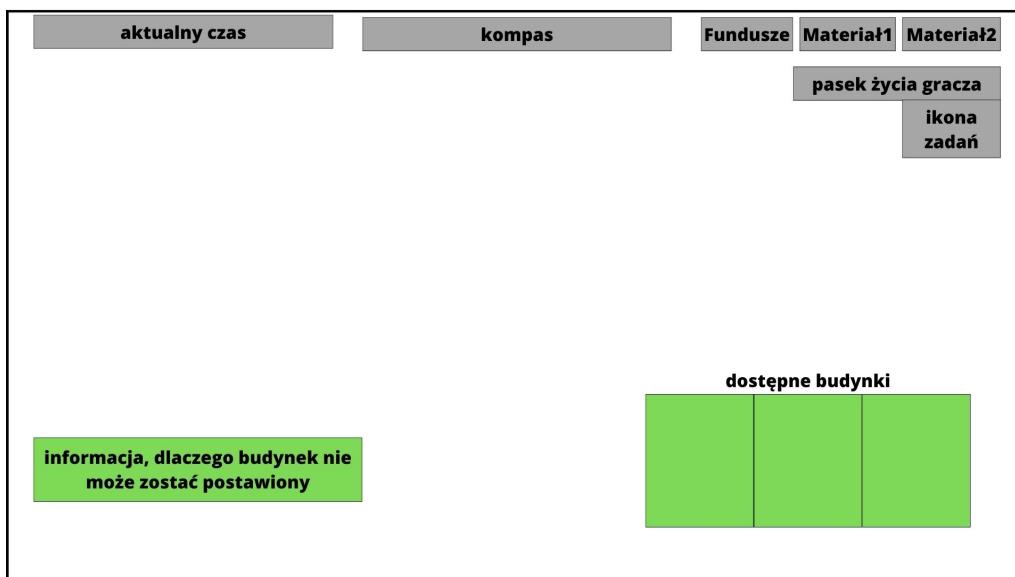
W naszej grze skupimy się jednak na tym, aby interfejs użytkownika zabierał jak najmniej miejsca. Dlatego też projekt zakłada, że poszczególne obiekty nie będą ze sobą połączone, a jedynie “dryfować” w przestrzeni. Jako ważny element tej części UI zawarty zostanie kompas, wzorowany na tym z gry *The Elder Scrolls V: Skyrim* (por. 2.8).

##### *4.5.2. Menu stawiania budynków*

Menu stawiania budynków będzie obrazować wiedzę i spostrzeżenia głównej postaci przy budowaniu nowej budowli. Docelowym miejscem ukazania się informacji będzie dół ekranu, aby interfejs nie zasłaniał użytkownikowi zbyt dużo przestrzeni. Graczowi ukaże się lista dostępnych budynków oraz odpowiednie komunikaty w wypadku, gdy nie można będzie dokonać zakupu, czy postawienia. Inspiracją do przedstawienia tych informacji zostanie rozwiązanie gry *Orcs must die!* (por. 2.5).



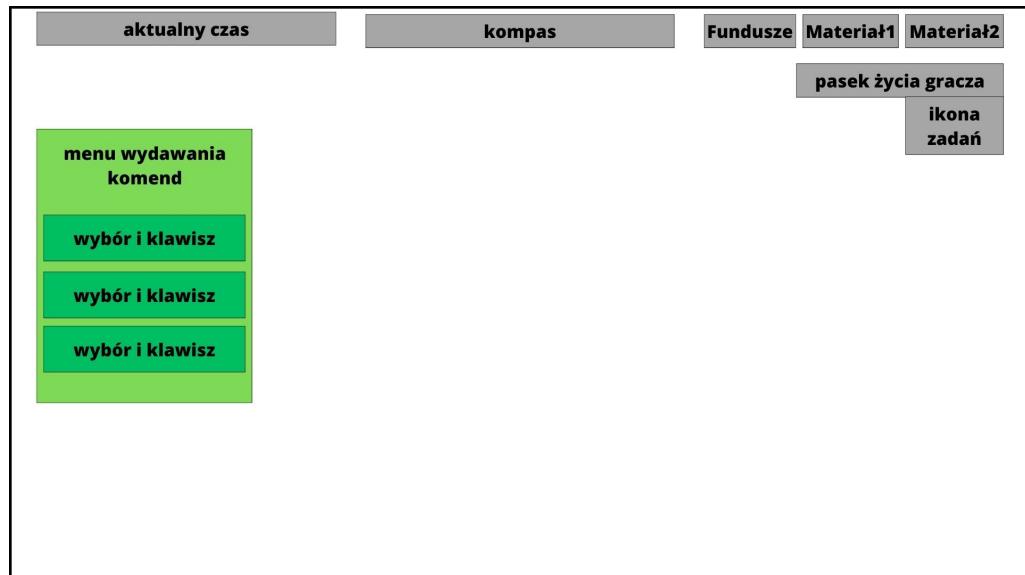
Rysunek 4.5: Projekt interfejsu podstawowego UI.



Rysunek 4.6: Projekt menu stawiania budynków.

#### 4.5.3. Menu wydawania komend

Po zdobyciu towarzyszy broni, kluczowe będzie udostępnienie mechaniki sterowania nimi. Zrealizowane to zostanie za pomocą menu wydawania komend. Projekt przewiduje wyświetlenie listy dostępnych komend i klawiszy, po których naciśnięciu, konkretna opcja zostanie wybrana. Źródłem pomysłu jest interfejs zaprojektowany w grze *Mount&Blade* (por. 2.7).



Rysunek 4.7: Projekt menu wydawania komend.

#### 4.5.4. Menu zapisu

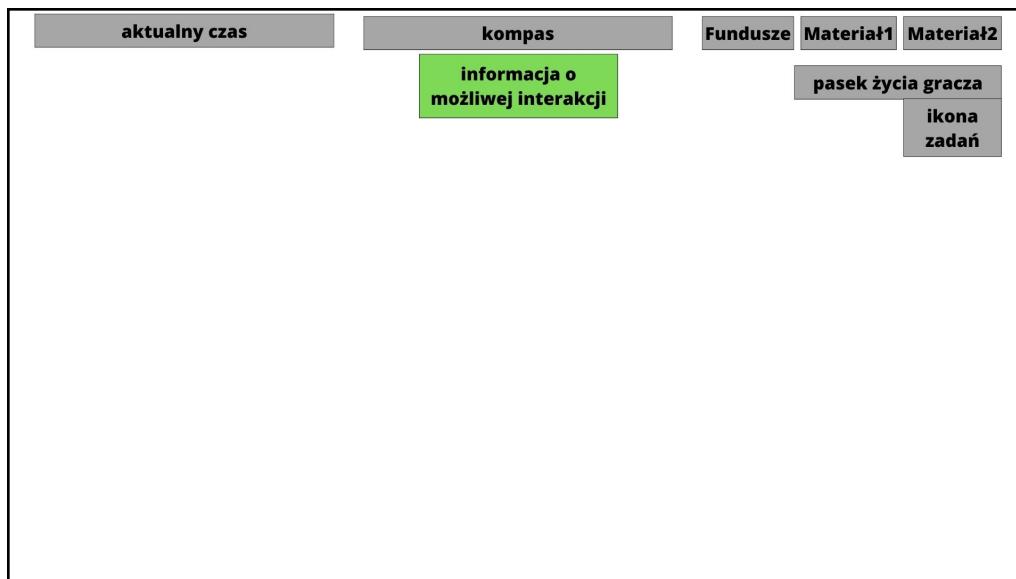
Aby umożliwić zapis gry przygotowane zostanie specjalne dla tego zadania menu, wyświetlane na środku ekranu. Najważniejszą informacją umiejscowioną na nim będzie nazwa pliku, aby gracz mógł później go odnaleźć.



Rysunek 4.8: Projekt menu zapisu.

#### 4.5.5. Informacja o możliwej interakcji

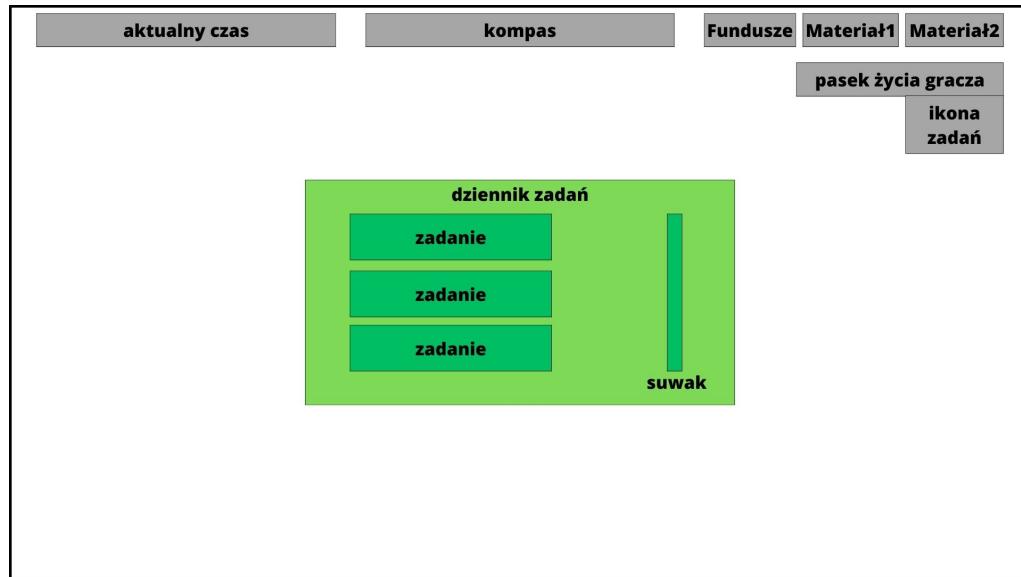
W programie przewidziane są postacie i przedmioty interaktywne. Aby zasygnalizować graczowi istniejącą możliwość podjęcia pewnej akcji, gra powinna mu wyświetlić stosowny komunikat. Wybranie widocznego miejsca będzie istotne, ponieważ grafika zniknie, jak tylko interakcja przestanie być możliwa. Tużże kluczowe będzie wybranie takiej lokalizacji, aby komunikat zwrócił uwagę gracza nawet, jeśli wyświetlony będzie tylko przez krótką chwilę. Za najwłaściwsze umiejscowienie wybrano środek górnej części ekranu, zaraz pod bazowym interfejsem.



Rysunek 4.9: Projekt grafiki informującej o możliwym rozpoczęciu rozmowy.

#### 4.5.6. Dziennik z zadaniami

Projektanci systemu zakładają, że niektóre postacie interaktywne będą mogły zlecić graczowi zadanie. Jeśli użytkownik przyjmie wiele zleceń, to zapamiętanie wszystkich szczegółów może stać się kłopotliwe. Dlatego projekt systemu przewiduje udostępnienie mechaniki dziennika, w którym pojawią się informacje o rozpoczętych zadaniach. Na przygotowanej grafice każde z nich będzie miało własne okno ze streszczeniem. Użytkownik będzie mógł je przeglądać posługując się suwakiem.



Rysunek 4.10: Projekt dziennika z aktualnie zaczętym i nieukończonym zadaniem.

#### 4.5.7. Ekran końca gry

Gracz, po wykonaniu pewnych kroków, może doprowadzić grę do stanu końcowego. W takim wypadku przewidziane jest wyraźne tego zasygnalizowanie przez program. Przewiduje się zaprojektowanie specjalnego ekranu końca gry. W tym celu użytkownikowi zostanie wyświetlona klarowna informacja o aktualnym stanie rzeczy oraz instrukcja, co zrobić dalej. Dodatkowo, aby wzmacnić przekaz, cały ekran zostanie przyciemniony.



Rysunek 4.11: Projekt ekranu końca gry.

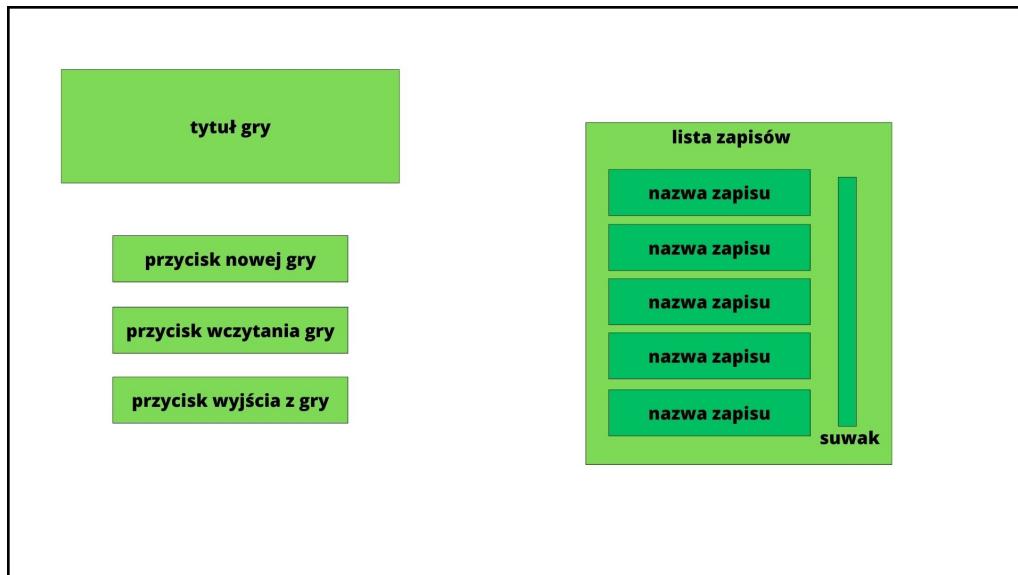
### 4.6. Menu główne (Zofia Sosińska)

Pierwsze co zobaczy użytkownik po włączeniu programu to menu główne, więc jego zadaniem będzie oddanie klimatu programu, umożliwienie rozpoczęcia rozgrywki oraz zamknięcie aplikacji. Projekt

zakłada, że menu główne udostępnii kluczowe funkcjonalności za pomocą przycisków:

- rozpoczęcia nowej gry,
- wczytania zapisanej gry,
- wyjścia z programu.

Przycisk wczytania gry będzie musiał pozwolić na wybranie pliku zapisu. W tym celu przewiduje się wyświetlenie listy z możliwością przewijania treści za pomocą suwaka. Jej elementami będą przyciski z nazwą zapisu, których naciśnięcie przeniesie użytkownika do konkretnego stanu gry.



Rysunek 4.12: Projekt menu głównego.

#### 4.7. Nawigacja (Zofia Sosińska)

Kluczowym dla gry założeniem jest ułatwienie graczowi wczucia się w realia świata, w którym się znajduje. Jako jeden z głównych warunków pogłębiania immersji uwypuklono brak implementacji mapy, na której gracz widziałby świat. W ten sposób nie upraszczamy mu poruszania się i odnajdywania lokacji tak, jak i człowiek w realnym świecie w czasach średniowiecznych nie kierował się zapisanymi na kartce kartograficznymi obrazami, ale własną i zdobytą od innych wiedzą o otaczającym go terenie.

Jedyną pomocą, jaką otrzyma gracz, będzie pasek obrazujący pole widzenia granej postaci. Pierwszą rolą narzędzia będzie pokazanie kierunku świata, który znajduje się w polu widzenia gracza. Zakładamy, że grana postać potrafi sama taką informację odczytać, chociażby z położenia Słońca.

Kolejną informacją na omawianym elemencie będzie miejsce, w którym znajduje się przeciwnik. Dotyczy to zarówno antagonistów widocznych w polu widzenia, jak i tych ukrytych za ścianą. Druga część będzie logicznie możliwa dzięki specjalnej umiejętności widzenia wrogów za przeszkodą zaimplementowanej dla postaci druida. Po przyłączeniu takiej osoby do drużyny użytkownik może poprosić przyjaciela o użyczenie mu swej mocy.

#### 4.8. Poruszanie postacią (Bogna Lew)

Podstawową mechaniką, którą będzie oferować prototyp gry, jest sterowanie postacią przez gracza. Chociaż nie jest to typowy element gier strategicznych czasu rzeczywistego, to zespół zdecydował się na

jego implementację w celu umożliwienia graczowi na silniejsze zanurzenie się w realia historyczne gry. Użytkownik będzie mieć do dyspozycji jedną postać, którą będzie bezpośrednio zarządzać. Umożliwi mu ona przede wszystkim eksplorację świata oraz walkę z przeciwnikami.

Model postaci będzie mieć ubrany pancerz i nie będzie posiadać żadnej broni. Jest to spowodowane faktem, że zgodnie z zamysłem gry gracz ma być dowódcą, a nie wojownikiem. "Model awatara gracza (czyli postać gracza) może mieć silny wpływ na jego zachowanie. [...] Jeśli postać gracza ma miecz, gracz oczekuje, że będzie mógł uderzyć nim w coś i stanąć do walki" [18].

Inspiracją dla mechaniki sterowania postacią jest gra *The Elder Scrolls V: Skyrim* (por. 2.9). Gracz będzie mógł przemieszczać się do przodu, do tyłu, na boki oraz na ukos. Sterowanie będzie możliwe za pomocą klawiszy W, A, S oraz D. Dodatkowo użytkownik będzie mieć możliwość obracania postaci, a tym samym zmiany kierunku, w który jest zwrócona poprzez przemieszczanie myszy. Ponadto gra udostępnia możliwość zmiany wysokości i kąta patrzenia kamery.

Kolejnym aspektem jest walka. Użytkownik będzie mógł wykonać atak poprzez naciśnięcie lewego przycisku myszy. Postać gracza będzie mogła wykonywać wyłącznie ataki wręcz. Ma to na celu zmotywowanie go do wynajmowania jednostek posługujących się bronią, a co za tym idzie - silniejszych od niego.

#### **4.9. Rozwój postaci gracza (Bartosz Strzelecki)**

System rozwoju postaci jest inspirowany serią gier *The Elder Scrolls* (por. 2.10) i będzie stanowił podstawowy element rozwoju toku rozgrywki. Użytkownik będzie rozwijał umiejętności swojej postaci poprzez wykonywanie czynności związanych z daną statystyką. Cechy, które gracz może rozwijać, korzyści, jakie z nich płyną i sposób ich rozwoju przedstawia tabela 4.1.

**Tabela 4.1:** Rozwijalne umiejętności gracza.

Statystyka	Sposób rozwoju	Korzyści
Siła	Własnoręczne pokonywanie wrogich jednostek	Większe obrażenia zadawane przez postać gracza podczas uderzeń
Przywództwo	Pokonywanie wrogich jednostek z użyciem pomocy przyjaznych wojowników	Większe obrażanie zadawane przez przyjazne jednostki
Charyzma	Pomaganie mieszkańcom świata	Przychylniejsze efekty podczas rozmów z postaciami niezależnymi

#### **4.10. Widzenie przez ściany (Bartosz Strzelecki)**

Do umiejętności wykorzystywanych przez gracza będzie należeć zdolność widzenia przeciwników oraz innych istotnych obiektów przez przeszkody. Gracz po naciśnięciu przycisku przez krótki okres będzie w stanie zobaczyć sylwetki przeciwników znajdującymi się w jego polu widzenia. Rozwiążanie

jest inspirowane wcześniej wspomnianą grą *Dead by Daylight* (por. 2.11). Po pojawienniu się, markery nie będą się poruszać za celem, lecz pozostać w tym samym miejscu przez czas trwania animacji.

#### **4.11. System dialogów (Bartosz Strzelecki)**

System dialogów jest podstawową metodą, którą gracz będzie wykorzystywał, aby pozyskać informacje o świecie oraz celach misji. Gracz może inicjować konwersacje z postaciami niezależnymi, po czym zostaną mu zaproponowane opcje sposobu prowadzenia rozmowy. W zależności od wybranych opcji dialogowych gracz może się spodziewać różnych konsekwencji. "Postacie niezależne (NPC) w grach są jednym z najbardziej złożonych i uniwersalnych sposobów pośredniego prowadzenia graczy, który może przybierać różne formy" [18].

Dialogue Editor autorstwa Grasshop Dev jest prostym narzędziem pozwalającym na szybkie dodawanie i modyfikację dialogów. Zawiera zestaw elementów ułatwiających wdrożenie systemu do projektu oraz udostępnia struktury danych wykorzystywanych do tworzenia interfejsu użytkownika. Podczas rozmowy z postaciami niezależnymi gracz będzie mógł pozyskać informację o geografii świata, możliwych zagrożeniach oraz zadaniach do wykonania. Podobne systemy występują w grach takich jak Pillars of Eternity oraz w grach z serii *Mass Effect* (por. 2.6).

#### **4.12. Mechanizm budowania (Bogna Lew)**

Inspiracją do implementacji tego mechanizmu jest gra *Warhammer 40,000: Dawn of War* (por. 2.3). Do pożądanych efektów, które ten tytuł zapewnia, należą walidacja terenu oraz wymuszanie poniesienia kosztów budowy. Dodatkowo wytwarzana gra będzie implementować proces budowania analogicznie jak w *Warhammer 40,000: Dawn of War*.

Najważniejszym aspektem implementowanego mechanizmu będzie walidacja terenu. Zostanie ona uzyskana poprzez wyświetlenie podglądu budowli w trakcie umiejscawiania konstrukcji na mapie. Jeśli miejsce, w którym gracz chce postawić budynek jest poprawne tzn. nie nachodzi na inne obiekty oraz teren jest odpowiedni, to pokazywany widok jest podświetlany na zielono, w przeciwnym razie - na czerwono. Jest to efekt, który wytwarzana przez zespół gra będzie zawierać.

Kolejnym pożdanym efektem jest konieczność poniesienia przez użytkownika kosztów wybudowania obiektu. W tym celu gra będzie monitorowała, czy gracz posiada wystarczającą ilość wymaganych zasobów, a w przypadku niespełnienia tych warunków - blokowała możliwość budowy wybranego obiektu.

Dodatkowo budowa obiektu nie może być natychmiastowa, gdyż nie jest to zgodnie z rzeczywistością. Dlatego podobnie jak w grze *Warhammer 40,000: Dawn of War* każdy budynek będzie musiał zostać wybudowany przez dedykowaną do tego postać. Będzie ona imitowała proces budowania i dopiero gdy ukończy to zadanie, dana budowla zacznie przynosić graczowi korzyści.

#### **4.13. Sterowanie jednostkami, podążanie za główną postacią (Zofia Sosińska)**

Na samym początku gry postać gracza pojawia się sama i jest jedynym obiektem, którym gracz może sterować. Kontroluje, gdzie postać idzie, jak walczy oraz z kim rozmawia. Z biegiem czasu gra będzie jednak naciskać na formowanie drużyny, ponieważ pokonywanie wielu przeciwników w pojedynkę będzie się stawało zbyt trudne. Pojawia się w takim momencie problem. Stworzenie mechaniki poruszania się i oddziaływania na otaczający świat dla jednej postaci wydaje się proste i intuicyjne, ale kierowanie wieloma osobami już nie. Bez wprowadzenia zmian, używając jednego sposobu dyrygowania wszystkimi

jednostkami tak samo, gra będzie prędko męczyć gracza. Dla czynności małoznaczących takich jak przemieszczenie drużyny w konkretne miejsce wprowadzi monotonię i czasochłonność. Każdą postać należy wybrać i przemieścić ją w konkretne miejsce. Kilka kliknięć przy jednej postaci jest akceptowalne, ale przy kilku wprowadzi to ogromne opóźnienia. Gracz byłby już wielokrotnie dalej. Jeszcze gorsze skutki pokazałyby się podczas walki. Szybkie przeskakiwanie pomiędzy postaciami podniosłoby zauważalnie trudność gry. Poruszanie się jedną postacią i zabijanie przeciwników nie ma sensu, gdy reszta drużyny jest bita i nie może się obronić, ponieważ gracz musi się przełączyć na inną postać, aby ta wykonała ruch. Stąd potrzeby jest algorytm odpowiadający za właściwe poruszanie się pobocznych postaci. Tworzona gra będzie dopuszczała małe, kilkunastoosobowe drużyny z przywódcą - postacią grywalną przez użytkownika - na czele. Podczas walki graczowi pokażą się możliwe do wydania polecenia oraz specyfikacja, jakiej grupy mają one dotyczyć. Za pomocą określonych klawiszy klawiatury będzie on mógł kontrolować zachowanie kompanów. Po rozwiązaniu problemu mechaniki sterowania jednostkami w walce nie można przeoczyć samego poruszania się oraz interakcji ze światem. Najprostszym rozwiązaniem będzie implementacja mechanizmu, według którego drużyna, po wykryciu znacznego przemieszczenia się przywódcy, sama będzie za nim podążać. Kompani nie będą też mieli opcji samodzielnej interakcji ze światem. Poza walką zostaną jedynie biernymi obserwatorami.

#### **4.14. Sztuczna inteligencja (*Bartosz Strzelecki*)**

W przypadku implementacji mechanizmów sztucznej inteligencji przeciwników będziemy się inspirować trybami kampanii w grach *Warcraft III* oraz *Starcraft II* (por. 2.2). Na mapie będą rozsiane punkty, w których będą pojawiać się przeciwnicy. Tak długo, jak drużyna gracza będzie się znajdować poza zasięgiem, wrogowie pozostaną nieaktywni. Aktywni przeciwnicy będą się zachowywać zgodnie z ich archetypem (klasą postaci) oraz pozostaną aktywni tak długo, jak drużyna gracza będzie w zasięgu. Zdezaktywowani przeciwnicy będą powracać do swojego oryginalnego stanu. Gracz będzie napotykał tego typu obozowiska przede wszystkim w trakcie eksploracji świata.

Będziemy wyróżniać trzy archetypy jednostek w zależności od sposobu walki (bliski zasięg, średni zasięg, daleki zasięg). Postacie walczące w bliskim zasięgu będą mieć na celu podejście w stronę najbliższego przeciwnika, aby wykonać atak. Jednostki średnio zasięgowe w momencie, w którym najbliższy przeciwnik jest odpowiednio daleko, będą wykonywać atak dystansowy, w przeciwnym wypadku będą się zachowywać tak jak jednostki walczące w zwarciu. Postacie dalekodystansowe będą wykonywać wyłącznie ataki dystansowych w kierunku najbliższego przeciwnika oraz uciekać, gdy ten podejdzie zbyt blisko.

## 5. IMPLEMENTACJA

Ten rozdział skupia się na zaprezentowaniu implementacji projektu w silniku Unity. Przedstawia on wykorzystane metodyki i narzędzia wykorzystane do osiągnięcia określonych wymagań, jak również zawiera fragmenty wykorzystanych algorytmów i zrzuty ekranu reprezentujące efekty pracy.

### 5.1. Interfejs Użytkownika (Zofia Sosińska)

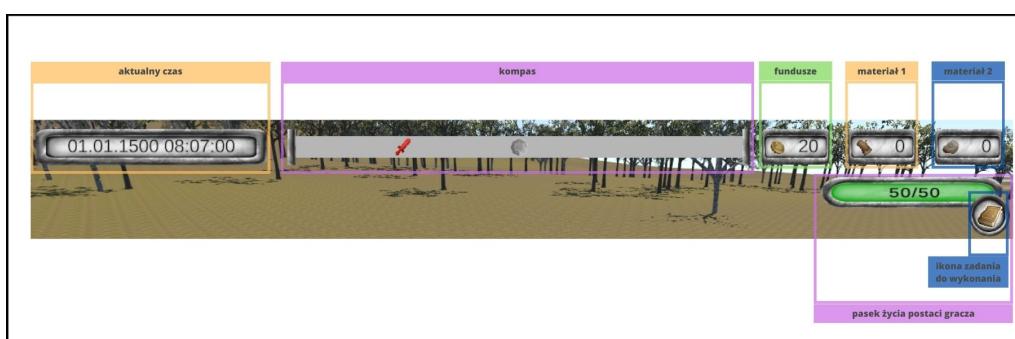
Interfejs użytkownika, jako uporządkowany i przejrzysty obraz wiedzy i możliwych opcji granej postaci odciąża użytkownika aplikacji, zdejmując z niego przymus pamiętania dokładnie każdej pojawiającej się informacji. Zabieg estetyczny, jakim jest przedstawienie suchych danych w postaci prostych obrazów, wyszczególniając to, co jest najważniejsze, sprawia, że rozgrywka staje się dla użytkownika prostsza, a co za tym idzie - przyjemniejsza.

Tak jak zaprojektowano, UI udostępnia interfejs podstawowy z zawsze widocznymi elementami oraz dynamicznie pojawiające się okna, wywoływane za pomocą konkretnych klawiszy. Wszystkie łączy surowy i prosty przewodni motyw graficzny, wykorzystujący także różnorodne obrazy dla urozmaicenia. Przy implementacji ważne także było, aby UI zabierał jak najmniej miejsca, jednocześnie podając jak najwięcej przydatnych informacji.

#### 5.1.1. Interfejs podstawowy

Interfejs podstawowy towarzyszy graczowi podczas całej rozgrywki. Skupia się on w górnej części ekranu. Jego zadaniem jest pomóc użytkownikowi w ogólnym odnalezieniu się w świecie. W tym celu są mu ukazane następujące informacje:

- aktualny czas w grze,
- położenie gracza względem stron świata oraz wrogów ukazane na kompasie,
- stan surowców i funduszów,
- stan zdrowia gracza,
- etap, na którym są przypisane graczowi zadania.



Rysunek 5.1: Implementacja paska z najważniejszymi informacjami o stanie gry: aktualnym czasie, posiadanych surowcach i funduszach, położeniu i otoczeniu gracza, zdrowiu postaci oraz ikona sygnalizująca, czy użytkownik ma do wykonania jakieś zadanie.

W tych statycznie umiejscowionych elementach interfejsu dynamicznie zmienia się ich treść. Na

zegarze czas zmienia się w ustalony sposób, dodając minutę w grze co 5 sekund upływające w realnym świecie. Na kompasie umiejscowienie zmieniają symbole stron świata i wskaźniki przeciwników, zależnie od ich pozycji, położenia gracza i strony, w którą główna postać patrzy. Fundusze rosną po wykonaniu zadania i otrzymaniu zapłaty oraz maleją, gdy opłacamy najemników, czy płacimy za budowle. Liczba posiadanych surowców zwiększa się, po zebraniu ich z ziemi, za to zmniejsza, gdy za nich pomocą budujemy budynek. Pasek zdrowia zmienia swoją wartość, gdy dostaniemy obrażenia, jednocześnie ukazując początkową, maksymalną wartość. Ikona zadania do wykonania ukazuje symbol czerwonego wykrzyknika, gdy gracz ma jakieś aktywne, nieskończone zadanie.

### 5.1.2. Menu stawiania budynków

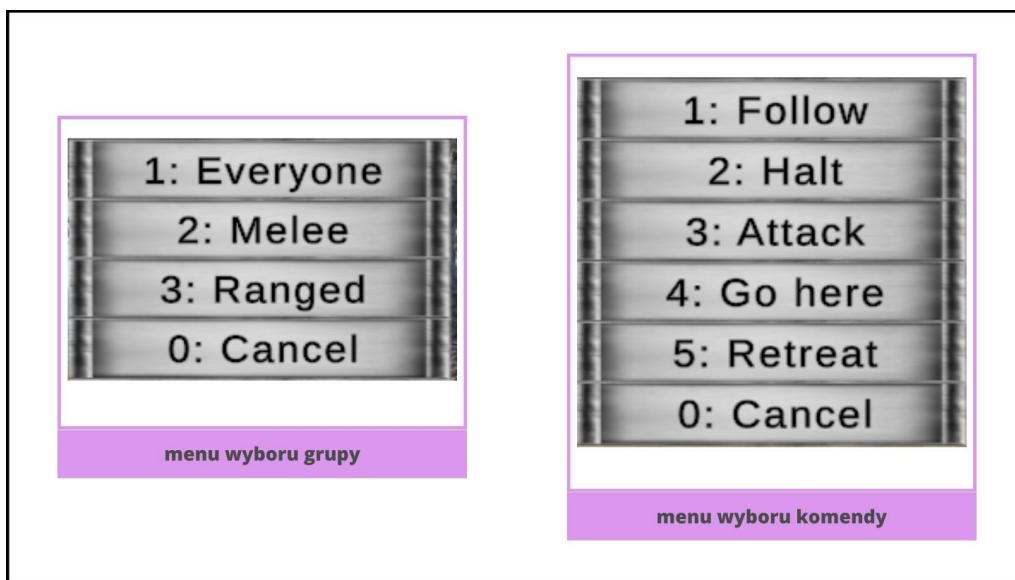
Typowa mechanika gier typu RTS, czyli budowanie budynków, ma specjalnie przygotowany interfejs, wyświetlany po wcisnięciu klawisza B. Ulokowany on został w dolnej części ekranu. Najważniejszym elementem menu stawiania budynków jest lista dostępnych budowli. Widnieją tam obrazy ukazujące każdą z nich, a gracz ma wgląd w ich szczegóły poprzez zmianę aktywnej opcji za pomocą prawej lub lewej strzałki. Wtedy wokół niej pokazują się także informacje dotyczące jej kupna, a po lewej stronie ekranu - informacja o ewentualnym nieprawidłowym umiejscowieniu. W wypadku, gdy zakup jest niemożliwy z powodu niewystarczającej liczby surowców, pojawi się stosowny komunikat. Menu zamyka się za pomocą klawisza Escape.



Rysunek 5.2: Implementacja menu stawiania budynków, na którym pokazane są możliwe do zbudowania budowle i szczegółowe informacje o ich dostępności.

### 5.1.3. Menu wydawania komend

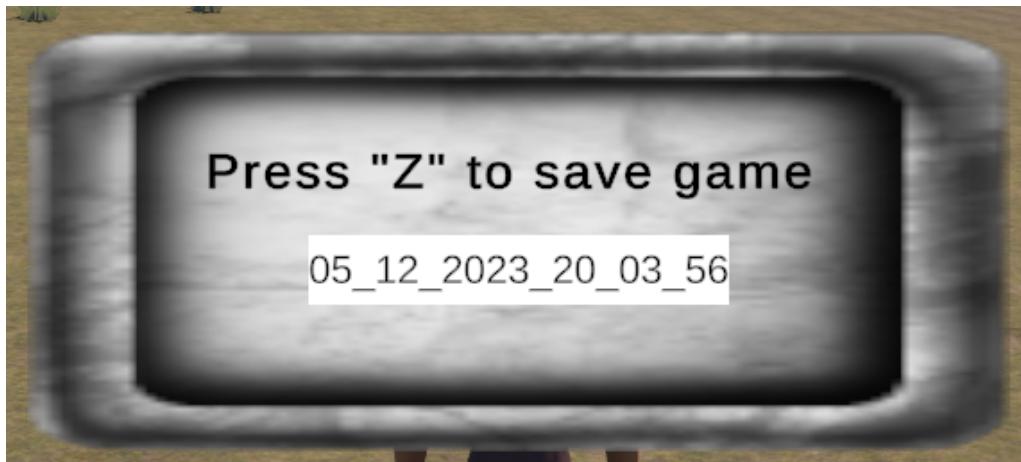
Gra umożliwia graczowi wykupienie usług najemników, jeśli dysponuje odpowiednimi funduszami. Gdy są oni już pod dowództwem użytkownika, może on im rozkazywać za pomocą menu wydawania komend. Po naciśnięciu klawisza Q pokazuje się lista dostępnych grup podwładnych, podzielonych według ich specjalizacji. Po wybraniu jednej z nich wylistowane zostaną możliwe komendy do wydania. W obu przypadkach gracz ma także możliwość anulowania.



Rysunek 5.3: Implementacja menu wydawania komend.

#### 5.1.4. Menu zapisu

Po uznaniu przez gracza, że nie chce stracić aktualnego stanu gry, może go permanentnie zapisać w pamięci urządzenia, na którym włączony jest program. Po naciśnięciu klawisza Z pojawia się menu zapisu. Gracz może odczytać z niego nazwę pliku, w którym zapisany zostanie stan gry. W menu znajduje się także informacja o tym, że poprzez ponowne naciśnięcie klawisza Z potwierdzi on zapis.



Rysunek 5.4: Implementacja menu zapisu.

#### 5.1.5. Informacja o możliwej interakcji

W świecie gry postać gracza nie jest odosobniona, co więcej może spotkać wiele różnorodnych osób, z którymi można porozmawiać. Interaktywni jednak są nie tylko ludzie, ale i surowce, które można zbierać. Jeśli możliwa jest interakcja, wyświetlana jest informacja o takim stanie rzeczy. Zaraz pod kompasem pojawia się grafika instruująca, że po naciśnięciu klawisza E klawiatury, postać gracza wykona opisaną czynność.



Rysunek 5.5: Implementacja grafiki informującej o możliwości rozpoczęcia rozmowy.



Rysunek 5.6: Implementacja grafiki informującej o możliwości podniesienia przedmiotu.

### 5.1.6. Dziennik z zadaniami

Niektóre postacie interaktywne mogą zlecić głównemu bohaterowi zadanie. Jeśli gracz się zgodzi, notatka o zadaniu zostaje zapisana w dzienniku. Można do niego zatrzymać wciskając klawisz J. Graczowi ukaże się tytuł sygnaлизujący główny cel zadania, streszczenie najważniejszych informacji oraz porada dotycząca mechanik gry np. "Wciśnij klawisz X, żeby stało się Y". To, że zadanie nie zostało jeszcze wykonane, sygnalizuje czerwony wykryzykniak w prawym, górnym rogu notatki. Gracz może zamknąć dziennik poprzez ponowne naciśnięcie klawisza J.



Rysunek 5.7: Implementacja dziennika z aktualnie zaczętym i nieukończonym zadaniem.

### 5.1.7. Ekran końca gry

Postać gracza ma określoną liczbę punktów życia, które może stracić podczas potyczek z przeciwnikami. Gdy spadną one do zera, gra kończy się, a główny bohater umiera. W takim wypadku graczowi pojawia się ekran końca gry, który informuje go o śmierci oraz o możliwości przejścia do menu głównego. Aby jeszcze bardziej uwypuklić zakończenie rozgrywki, cały ekran zostaje przyciemniony.



Rysunek 5.8: Implementacja menu zapisu.

## 5.2. Menu główne (Zofia Sosińska)

Przed startem właściwej gry, użytkownikowi ukazuje się menu główne. Pełni ono funkcję reprezentatywną, więc ważne jest, aby współgrało ono z głównym programem. Styl okna jest prosty i surowy, stawiając na skalę szarości przy doborze kolorów. Nawiązuje on do stylu interfejsu użytkownika występującego w trakcie rozgrywki. Sprawia to, że gracz jeszcze przed zanurzeniem się w świat gry, ma jego przedsmak.

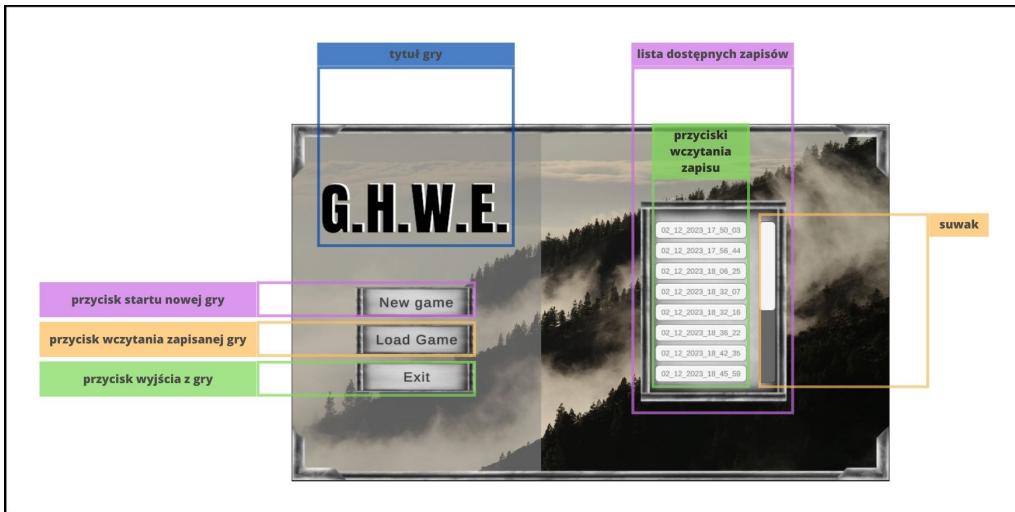
Zgodnie z projektem menu główne udostępnia trzy kluczowe funkcjonalności:

- rozpoczęcie nowej gry,
- wczytanie zapisanej gry,
- wyjście z programu.

Przycisk nowej gry przenosi użytkownika do momentu startowego rozgrywki. Progres gry jest równy零, a wszystkie zadania dopiero czekają na wykonanie.

Po kliknięciu przycisku odczytu pojawia się lista zapisów, które w przeszłości użytkownik zdecydował się permanentnie przechować w pamięci urządzenia, na którym włączony został program. Użytkownik może ją przeglądać ruszając suwakiem po prawej stronie. Po wybraniu jednej z opcji zostaje przeniesiony do świata gry o stanie takim, jaki został wczytany z pliku.

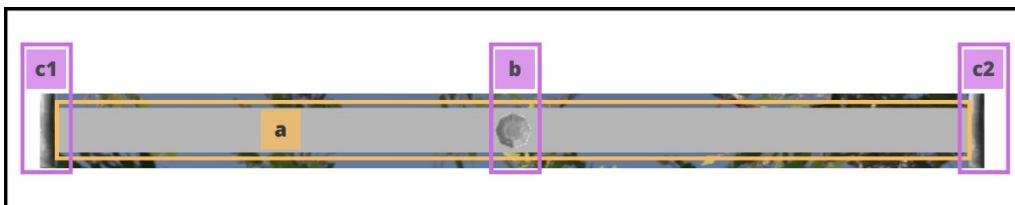
Ostatni przycisk udostępnia funkcjonalność wyjścia z gry. Po jego naciśnięciu użytkownik całkowicie wyłącza program.



Rysunek 5.9: Implementacja menu głównego.

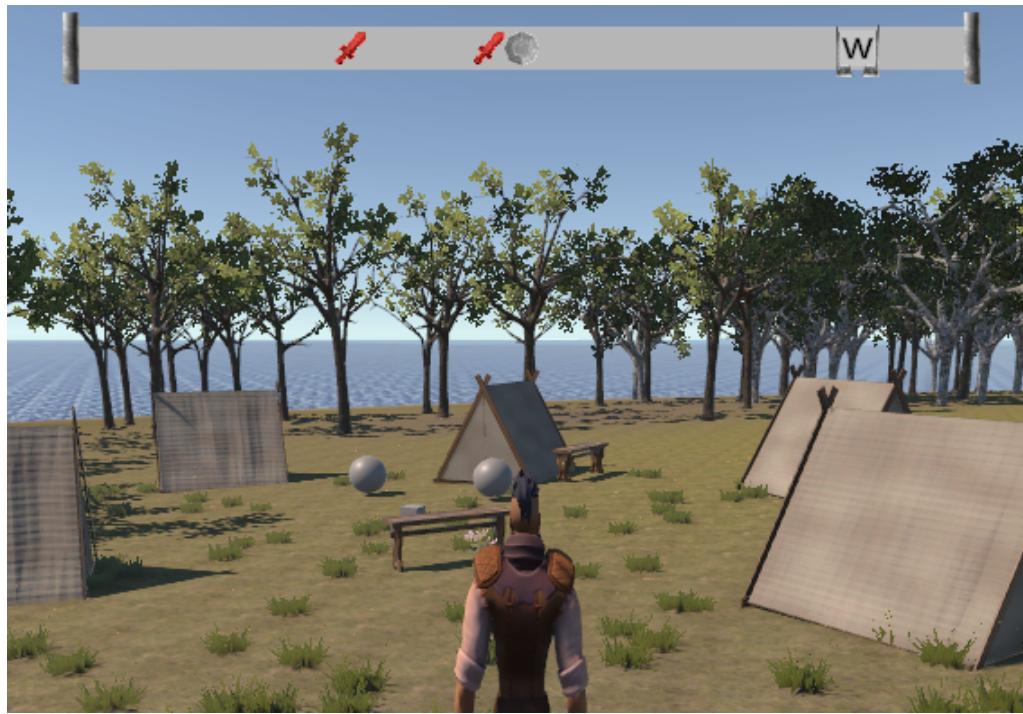
### 5.3. Zasady działania kompasu (Zofia Sosińska)

Projekt kompasu jest na tyle prosty, aby nie przytłoczyć gracza nadmierną liczbą bodźców. Składa się z horyzontalnego, jednolitego paska, na którym wyświetlane są najważniejsze informacje o otoczeniu, symbol ośmiokąta, wskazujący na przestrzeń znajdującą się centralnie przed bohaterem oraz z bocznych pasków, wyróżniających końce narzędzia.



Rysunek 5.10: Rozpiska elementów: a. główny pasek, b. symbol środka, c1., c2. paski końców kompasu.

Ikony wyświetlane na kompasie przedstawiają najważniejsze informacje w polu widzenia gracza, czyli stronę świata, w kierunku której jest on zwrócony, oraz przeciwników. Poniżej przedstawiono kod odpowiedzialny za wyznaczenie pozycji symbolu na omawianym narzędziu. Wejściem jest komponent RectTransform symbolu przypisanego danemu obiektemu oraz jego położenie. Po obliczeniu wektora prowadzącego do uzyskania np. pozycji wroga, wyznaczany jest kąt, o jaki musi się obrócić. To jest przeliczane na pozycję na kompasie i jeśli obiekt znajduje się w polu widzenia postaci gracza, to jest wyświetlany odpowiedni symbol.



**Rysunek 5.11:** Wizualizacja przypadku, w którym gracz patrzy centralnie na obozowisko wrogów. Na przeciwnie oraz po lewej stronie znajdują się przeciwnicy, co jest zasygnalizowane na kompasie za pomocą symboli mieczy. Znajduje się na nim także informacja, że bohater jest lekko odchylony od Zachodu.

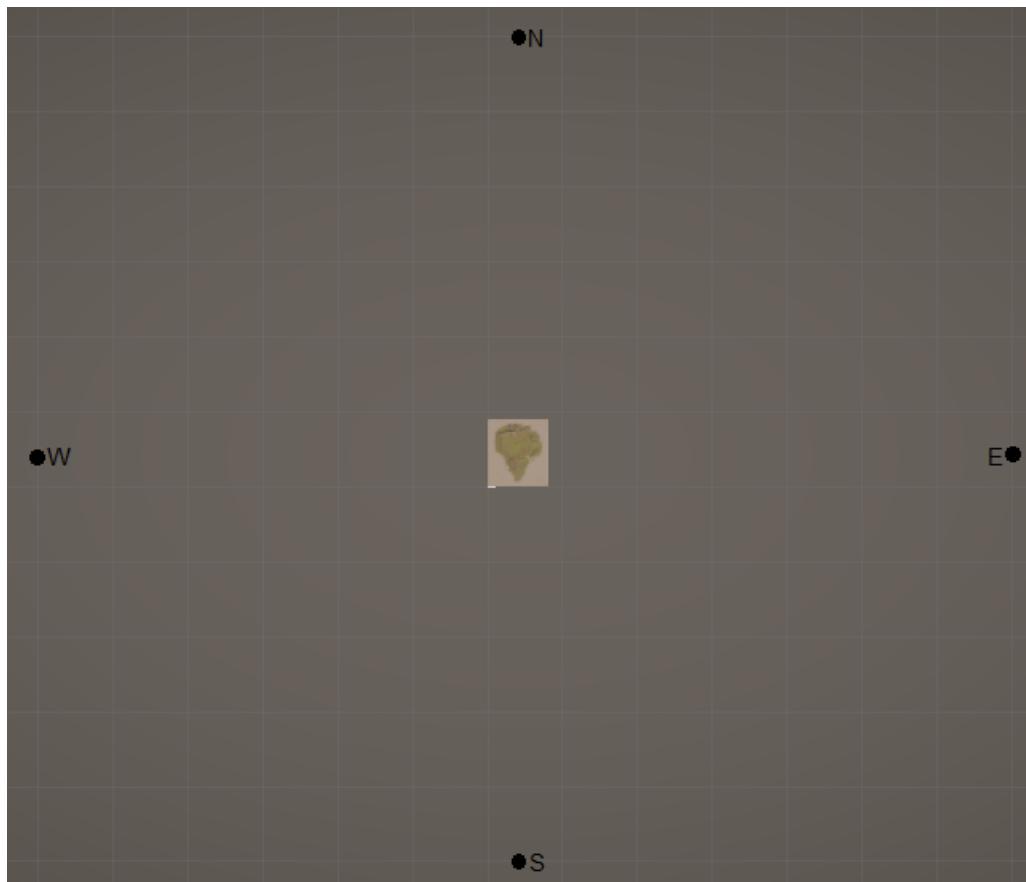
```

1 void SetMarkerPosition(RectTransform markerTransform, Vector3 worldPosition)
2 {
3     Vector3 dirToTarget = worldPosition - CameraTransform.position;
4     float angle = Vector2.SignedAngle(new Vector2(dirToTarget.x, dirToTarget.z),
5                                         new Vector2(CameraTransform.transform.
6                                         forward.x, CameraTransform.transform.forward.z));
6     float compassPositionX = Mathf.Clamp(
7         angle / Camera.main.fieldOfView, -1, 1);
8     if (compassPositionX == 1 || compassPositionX == (-1))
9     {
10         markerTransform.anchoredPosition = new Vector2(0, 100);
11     }
12     else
13     {
14         markerTransform.anchoredPosition = new Vector2(
15             compassBarTransform.rect.width / 2 * compassPositionX, 0);
16     }
17 }
18 }
```

**Listing 5.1:** Fragment kodu odpowiedzialny za ustawienie symbolu na pasku kompasu.

Problem lokalizacji stron świata pojawił się, gdy nieprawidłowo wyświetlały się symbole po użyciu najprostszego rozwiązania. Na przykład dla Północy było to pobranie wartości od `Vector3.forward`, co jest skrótnym zapisem `Vector3(0, 0, 1)`. Lewy dolny róg mapy jest położony w początku układu współrzędnych, co oznacza, że wymagane jest przesunięcie, aby prawidłowo zasymulować strony świata. Północ i Południe zostały przesunięte do połowy szerokości, a Wschód i Zachód - długości mapy. Każde z nich zostało oddalone o 60000 jednostek, co pozwala na wiarygodną symulację stron świata.

na kompasie.



Rysunek 5.12: Rozmieszczenie zasymulowanych stron świata.

Wykrycie wrogich jednostek znajduje się w metodzie Start(). Każdemu obiekowi jest przypisywany symbol czerwonego miecza i w zależności od położenia wroga, obrazek wyświetlany jest odpowiednio na kompasie.

```
1 void SetPositionOfEnemies()
2 {
3     foreach (
4         var e in enemiesOnMap.Zip(enemiesOnUI, (x, y) => new {enemyOnMap = x,
5                                         enemyOnUI = y }))
6     {
7         SetMarkerPosition(e.enemyOnUI.GetComponent<RectTransform>(),
8                           e.enemyOnMap.transform.position);
9     }
10 }
```

Listing 5.2: Fragment kodu odpowiedzialny za połączenie wrogich obiektów na mapie z symbolami wyświetlanymi na kompasie.

#### 5.4. Kontroler postaci (Bogna Lew)

W trakcie rozgrywki istotnym aspektem wpływającym na jakość jest mechanizm sterowania postacią. Z tą mechaniką gracz ma bezpośredni kontakt, ponieważ to właśnie za jej pomocą może eksplorować świat.

Do implementacji tego mechanizmu zainspirowaliśmy się grą *Skyrim* (por. 2.9). Postać jest sterowana za pomocą klawiszy W, A, S oraz D, natomiast jej rotacja oraz obrót kamery jest kontrolowany przez mysz. Dodatkowo gracz może wykonywać ataki poprzez naciśnięcie lewego przycisku myszy.

Pierwszy element kontrolera odpowiada za przemieszczanie się postaci. Do tego wykorzystuje komponent Input Manager, w którym zostały odwzorowane odpowiednie klawisze dla każdej z osi, wzdłuż której gracz może się przemieszczać. Na tej podstawie wyznaczane jest faktyczne przesunięcie postaci względem kierunku, w którym jest zwrócona oraz modyfikowane jest jej położenie. Dodatkowo ten komponent odpowiada za wyznaczenie prędkości, z jaką gracz się porusza. Domyslnie postać przemieszcza się tempem chodu, jednakże jeśli gracz przytrzyma lewy klawisz Shift, to zacznie się poruszać biegiem.

Ten element dodatkowo odpowiada za wykrywanie, czy gracz chce wykonać atak. Przekazuje on następnie tę informację do komponentu odpowiedzialnego za określenie powodzenia ataku oraz wywołanie odpowiedniego mechanizmu informowania przeciwnika o odniesionych obrażeniach.

Druga część kontrolera odpowiada za ustawienie odpowiedniej animacji. Udostępnia metody umożliwiające uruchomienie animacji ataków, odniesienia obrażeń, śmierci lub przemieszczania się. Ostatnia z nich wykorzystuje wartości zwrotu wzdłuż obu osi oraz prędkość, które są wyznaczane w poprzednim komponencie. Na ich podstawie definiuje kolejne części nazwy animacji, po czym, jeśli jest inna niż aktualnie wyświetlna, uruchamia ją.

Kolejny komponent nadzoruje rotację postaci oraz co za tym idzie - kamery. Analogicznie wykorzystywany jest Input Manager, jednak w tym przypadku przechytywane jest przesunięcie myszy. Komponent udostępnia graczu możliwość sterowania kierunkiem, w którym postać patrzy, a co za tym idzie - względem którego się przemieszcza. Jednocześnie następuje dostosowanie położenia kamery tak, aby zawsze patrzyła w tym samym kierunku co postać gracza. Ponadto komponent umożliwia zmianę wysokości oraz kąta patrzenia kamery. Dzięki temu gracz może dokładniej zobaczyć, co znajduje się nad oraz tuż przed nim. Obrazują to rysunki 5.13 oraz 5.14.



Rysunek 5.13: Widok po przesunięciu kamery w górę.

Ostatni komponent odpowiada za kontrolowanie przybliżania kamery. W tym celu w Input Managerze zostało odwzorowane kółko myszy. Zwrócona przez system wartość jest wykorzystywana do obliczenia odległości kamery od postaci przy zachowaniu ustalonych przez grę ograniczeń. Dodatkowo komponent dokonuje również automatycznego przybliżania, jeśli wskutek przemieszczania się postaci,

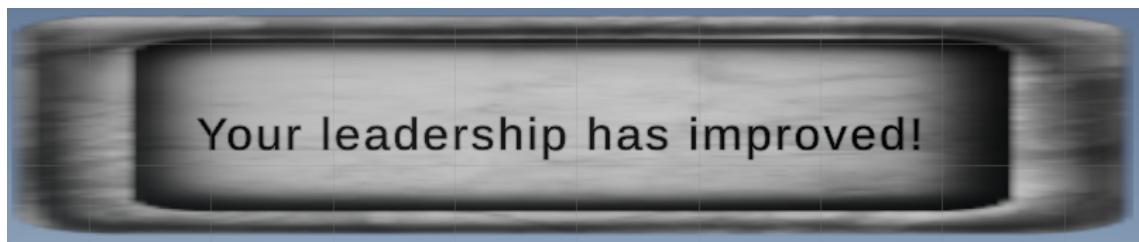


Rysunek 5.14: Widok po przesunięciu kamery w dół.

pomiędzy nią, a kamerą miałaby się znaleźć przeszkoda. Dzięki temu nie zostanie zasłonięty widok tego co się dzieje wokół postaci. Odsunięcie się od przeszkody zaskutkuje oddaleniem kamery do poprzedniej odległości.

### 5.5. Rozwój postaci gracza (Bartosz Strzelecki)

W implementacji systemu rozwoju postaci zastosowano system, w którym gracz zdobywa doświadczenie związane z daną statystyką zgodnie z tabelą 4.1. Po osiągnięciu pewnego progu, który jest liniowo zależny od dotychczasowego poziomu danej umiejętności, gracz uzyskuje korzyści ze zwiększonego poziomu statystyki. Po poprawieniu umiejętności postaci gracza na ekranie jest pokazywany odpowiedni komunikat (rys. 5.15) informujący użytkownika o nowo nabytych zdolnościach.



Rysunek 5.15: Komunikat informujący gracza o zdobyciu kolejnego poziomu umiejętności.

### 5.6. Widzenie przez horyzont (Bartosz Strzelecki)

Efekt został osiągnięty poprzez zmodyfikowanie potoku renderowania w taki sposób, że w zależności od wartości w buforze głębi jest wykorzystywany inny shader. W tym przypadku, jeżeli sfera jest przysłonięta przez ścianę, jest ona narysowana, a w przeciwnym wypadku jest uruchamiany pusty shader. "Unity domyślnie sortuje obiekty na podstawie odległości od kamery. Tak więc w miarę zbliżania się obiektu do kamery, będzie on rysowany nad wszystkimi obiektami znajdującymi się dalej od kamery. W większości przypadków sprawdza się to dobrze podczas tworzenia gier, ale znajdują się sytuacje, w których będziemy chcieli mieć większą kontrolę nad sortowaniem obiektów na scenie. Używając bloku Tags\\{\\} możemy kontrolować to sortowanie. Unity udostępniło nam kilka domyślnych kolejek renderowania, z których

każda ma unikalną wartość, którą kieruje Unity, kiedy należy narysować obiekt na ekranie. Te wbudowane kolejki renderowania nazywają się Background, Geometry, AlphaTest, Transparent i Overlay.” [19]. Wykorzystując ten mechanizm możliwe jest uzyskanie efektu widzenia przez postać gracza poprzez horyzont. Uzyskujemy przez to efekt podobny do tego zaimplementowanego w grze *Dead by Daylight* (por. 2.11).

Po naciśnięciu przycisku E następuje zagranie animacji opisanej wzorami

$$w(t, offset) = 1.1 \times 2.1^{-\left(\frac{(\sin(t)+1-0.4-offset)^2}{0.02}\right)} \quad (5-1)$$

oraz

$$w(t, 0) - w(t, -0.2) + w(t, -1) - w(t, -1.2) \quad (5-2)$$

Od podanych funkcji zależy przeźroczystość, jak i natężenie efektu Fresnela.

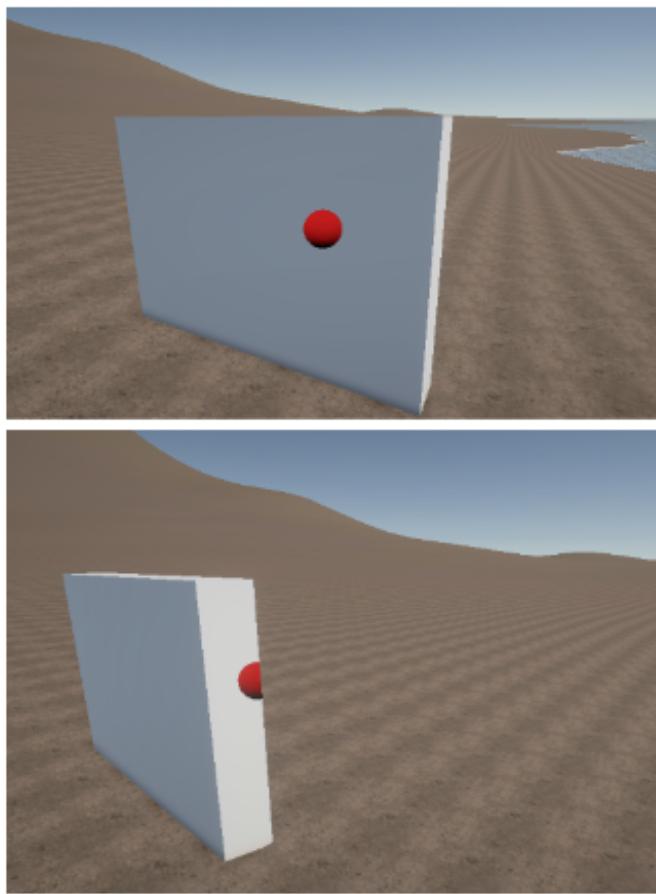


**Rysunek 5.16:** Wykres przedstawiający funkcję opisującą natężenie efektu w animacji markera.

```

1 fixed4 frag (v2f i) : SV_Target
2 {
3     float t = 6.2 * _Progress - 0.6;
4     fixed4 pattern = tex2D(_PatternTex, i.uv + _Speed * t);
5     float fresnelInfluence = dot(i.worldPos, i.viewDir);
6     float saturatedFresnel = saturate(1 - fresnelInfluence);
7
8     float g = w(t, 0) - w(t, -0.2) + w(t, -1) - w(t, -1.2);
9     float4 color = pow(saturatedFresnel, g * _FresnelPow) * (_Color * _ColorIntensity) *
    pattern;
10    color.a *= dot(i.worldPos, i.viewDir);
11    return color;
12 }
```

**Listing 5.3:** Fragment shadera odpowiedzialny za animację.



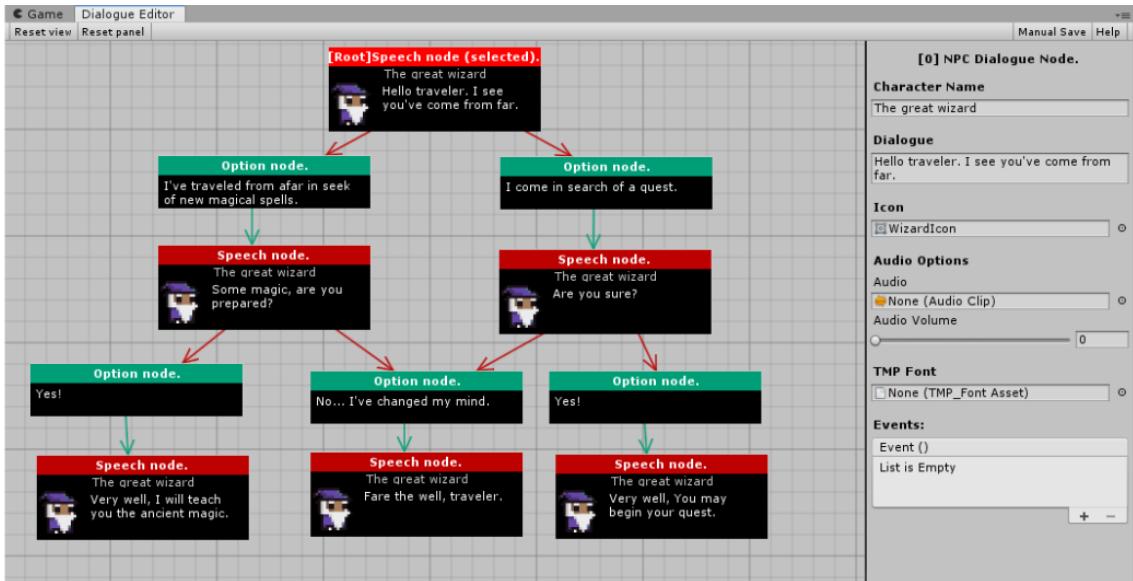
Rysunek 5.17: Zachowanie programu cieniąjącego w przypadku przesłaniania markera przez przeszkody.

### 5.7. System dialogów (*Bartosz Strzelecki*)

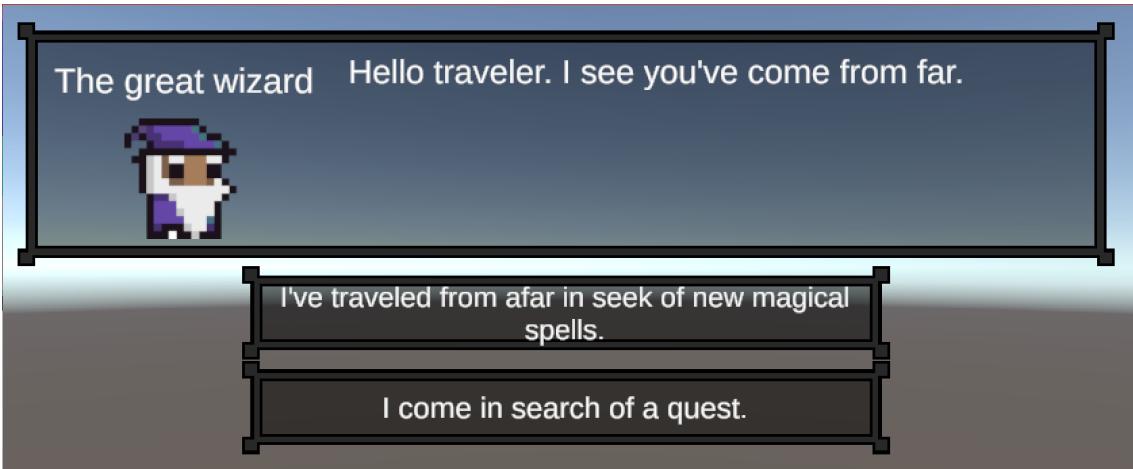
W naszej implementacji konwersacje składają się z dwóch rodzajów węzłów. Jednego odpowiedzialnego za wypowiedzi postaci niezależnych oraz drugiego pozwalającego na podjęcie przez gracza decyzji. Dodanie nowej konwersacji odbywa się poprzez stworzenie obiektu z przypisanym komponentem NPC Conversation. Wywołanie dialogu można osiągnąć poprzez wykorzystanie metody

```
ConversationManager.Instance.StartConversation(/*NPCConversation*/);
```

System pozwala na łatwe dostosowanie elementów interfejsu użytkownika odpowiedzialne za wyświetlenie dialogów, aby jak najlepiej wpasować się w styl graficzny gry.



Rysunek 5.18: Przykładowa konwersacja z wykorzystaniem Dialogue Editor.



Rysunek 5.19: Przykładowe okno dialogowe widziane z perspektywy gracza.

## 5.8. Mechanizm budowania (Bogna Lew)

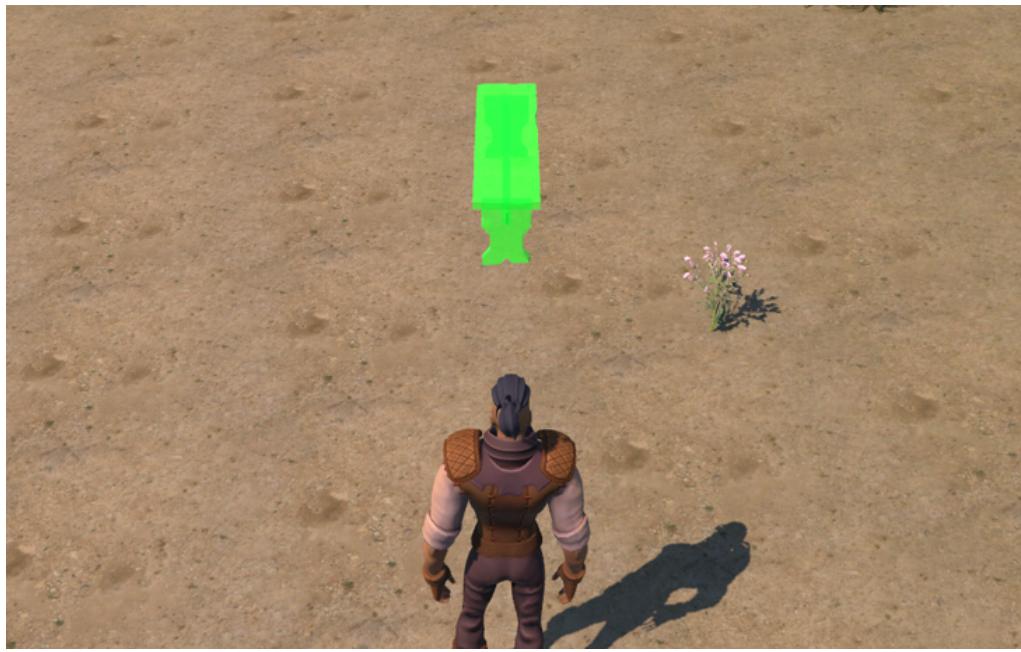
Opracowany mechanizm umożliwia graczu na przełączenie się w tryb budowania poprzez naciśnięcie klawisza B, który od razu wyświetli podgląd bazowego obiektu. Widok budynku przemieszcza się przed postacią oraz odpowiednio obraca się razem z nią. Efekt poruszania się podglądu został uzyskany za pomocą poniższego wzoru:

$$\begin{cases} x = r \times \sin(\alpha) \\ y = 0 \\ z = r \times \cos(\alpha) \end{cases} \quad (5-3)$$

gdzie  $x$ ,  $y$  i  $z$  to współrzędne odpowiednio względem osi X, Y i Z,  $r$  to odległość środka obiektu od postaci gracza, natomiast  $\alpha$  to kąt o jaki jest on obrócony względem osi Y.

Podgląd budynku będzie widoczny do czasu, aż gracz go umieści naciskając prawy przycisk myszy bądź wychodząc z trybu edycji naciskając klawisz Escape. Jeżeli widok budowli znajduje się w poprawnym do umieszczenia miejscu to jest on podświetlany na zielono, w przeciwnym razie - na czerwono.

Wybudowanie powoduje przywrócenie bazowych kolorów obiektu oraz włącza wykrywanie kolizji z nim, dzięki czemu budynek poprawnie oddziałuje z otaczającym go środowiskiem.



Rysunek 5.20: Przykład podglądu w poprawnym umiejscowieniu.

Do weryfikacji poprawności umiejscowienia budynku wykorzystano mechanizmy wyzwalacza (ang. *trigger*) oraz rzucania promienia (ang. *raycasting*). Pierwszy z nich polega na wykrywaniu czy obiekt znalazł się w obszarze kolizji bez brutalnego zatrzymywania go. Oznacza to, że może on przeniknąć przez inny element bez widocznych konsekwencji, jedynie wysyłając sygnał, że dana sytuacja nastąpiła. Druga metoda natomiast polega na wypuszczeniu promienia o pewnej długości w zadanym kierunku i sprawdzeniu, czy z czymś się zderzył.

Wyzwalacz został zastosowany do wykrywania kolizji z innymi obiektami na mapie, takimi jak postacie, czy elementy scenerii niebędące terenem. W tym celu wykorzystano metody `OnTriggerEnter()` i `OnTriggerExit()`, które są wywoływane odpowiednio gdy, dany obiekt znalazł się w obszarze kolizji innego elementu, bądź go opuścił. Każda z nich odpowiednio zmienia poprzez zwiększenie bądź zmniejszenie wartości licznika kolizji obiektu. Jeżeli jego wartość jest różna od zera, tzn. obiekt z czymś koliduje, to umiejscowienie jest uznawane za niepoprawne.

Drugi z mechanizmów został wykorzystany do sprawdzenia nachylenia podłoża. Efekt został osiągnięty poprzez rzucenie promienia o niewielkiej długości z każdego z dolnych narożników obiektu i zliczeniu ile z nich wykryło kolizję z ziemią. Brak wykrycia kolizji można zinterpretować jako zapadanie się bądź zbyt mocne lewitowanie danego narożnika nad powierzchnią ziemi. Z tego powodu przyjęto, że jeśli co najmniej trzy z nich zderzyły się z podłożem, to umiejscowienie jest poprawne.

Dodatkowo metodę rzucania promienia wykorzystano do wykrycia kolizji z drzewami. Obiekty te są traktowane przez silnik jako część terenu, przez co metoda wykorzystująca wyzwalacz pomija je. Dlatego w celu wykrywania kolizji z nimi zdecydowano się na rzucenie dwóch grup promieni biegących w głąb obiektu z naprzeciwległych krawędzi prostopadłych do osi Y. Każdy z promieni biegnie w połowie wysokości obiektu i sięga przeciwej ściany. Tak skomplikowany układ ma na celu zapewnienie poprawności wykrywania kolizji w przypadku, gdy początki promieni znajdują się wewnątrz kolidującego obiektu. Jeśli którykolwiek z nich zderzy się z drzewem, to wybrane miejsce jest uznawane za niepoprawne.



Rysunek 5.21: Przykład podglądu w niepoprawnym umiejscowieniu.



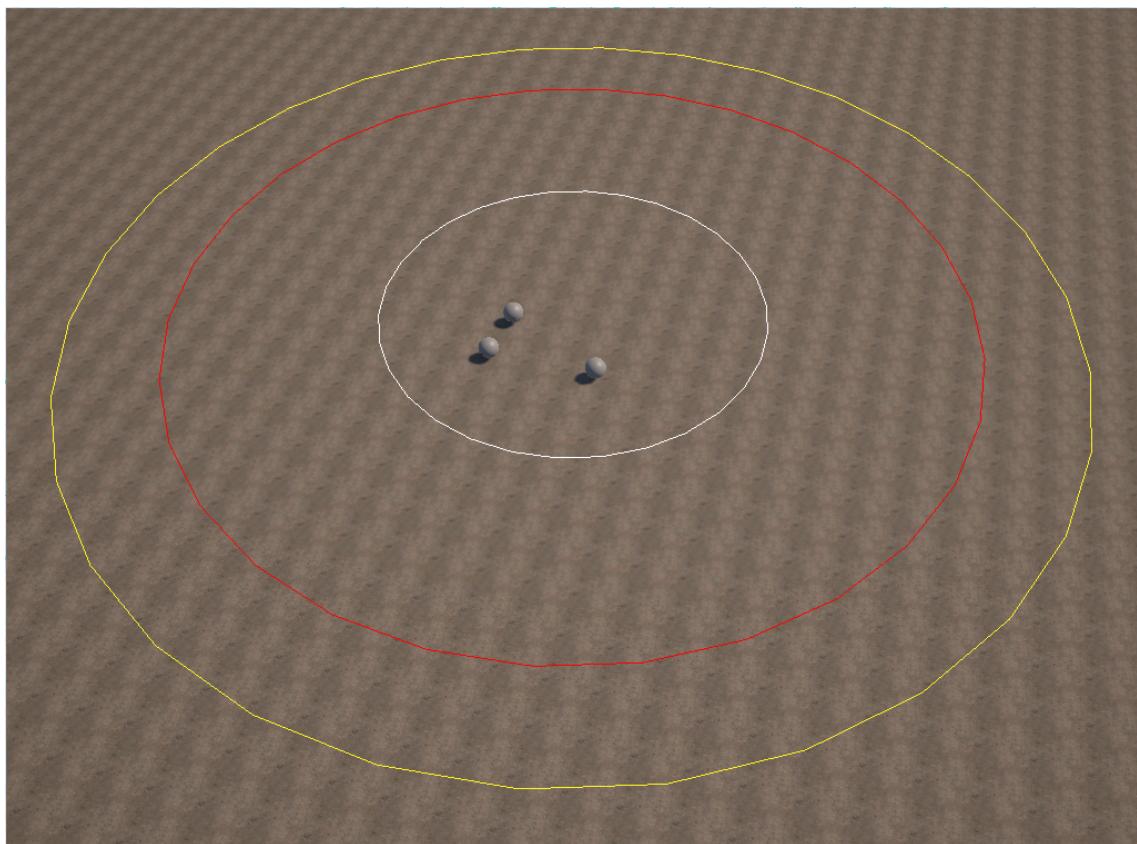
Rysunek 5.22: Widok z góry na siatkę promieni służących do wykrywania kolizji z drzewami.



Rysunek 5.23: Widok z boku na siatkę promieni służących do wykrywania kolizji z drzewami.

## 5.9. Sztuczna inteligencja (Bartosz Strzelecki)

Nawigacja przeciwników została zrealizowana poprzez wbudowany w silnik Unity system NavMesh. Pozwala on na łatwe wyznaczenie powierzchni, po której mogą poruszać się postacie niekontrolowane przez gracza oraz realizuje zadanie wyznaczania ścieżki dla tych postaci.



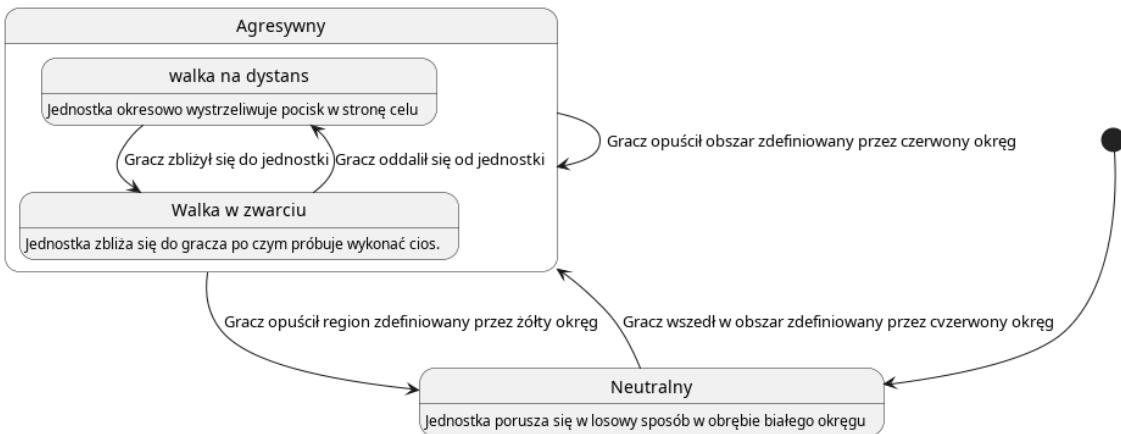
Rysunek 5.24: Obraz przedstawia zasięgi odpowiednich regionów.

Przeciwnicy są kontrolowani poprzez jeden obiekt przydzielający cele każdemu przypisanemu wrogowi. W normalnym trybie wrogowie poruszają się w sposób losowy w obrębie wyznaczonej przestrzeni (biały okrąg na rys. 5.24). Kiedy przyjazne jednostki znajdą się w wystarczającej odległości (czerwony okrąg na rys. 5.24), przeciwnicy obiorą sobie za cel jedną z nich. Po opuszczeniu przez drużynę gracza wyznaczonego obszaru (żółty okrąg na rys. 5.24) wrogowie wracają do poruszania się w sposób losowy w obrębie białego okręgu.

### 5.9.1. Mechanika zachowania klas jednostek

#### Jednostki walczące w zwarciu

Zachowanie jednostek bliskiego zasięgu polega na wybraniu przeciwnika będącego w czerwonym obszarze (rys. 5.24), który ma najwyższą wartość priorytetu obliczonego między innymi na podstawie odległości, ilości innych atakujących przeciwników oraz klasy celu. Jednostka posiadająca cel ataku zmierza w jego kierunku, obierając najkrótszą drogę. Po dotarciu do celu podróży jednostki biorące udział w walce losują naprzemiennie liczby, od których zależy wynik walki, jak i aktualnie wykorzystywana animacja.



**Rysunek 5.25:** Diagram przedstawiający przepływ stanów dla sztucznej inteligencji przeciwników.

### Jednostki walczące na dystans

Takie jednostki wystrzeliwują w stronę przeciwników pociski, których celność jest reprezentowana przez dwie strefy (rys. 5.26). Jedną oznaczającą 50% celności, czyli rozrzut, który będzie posiadała połowa pocisków, oraz drugą oznaczającą maksymalny rozrzut. Ten mechanizm pozwala na łatwe za-modelowanie celności ze względu na odległość, wielkość celu oraz na osłony, za którymi może stać wroga jednostka.



**Rysunek 5.26:** Reprezentacja graficzna celowania widziana z perspektywy jednostki dalekodzięsięowej. Czerwony okrąg reprezentuje obszar, w którym znajdzie się 50% pocisków. Żółty obszar pokazuje maksymalny rozrzut.

## Jednostki hybrydowe

Obejmują zachowanie dwóch powyższych klas jednostek w zależności od odległości od przeciwnika. W sytuacji, w której wroga jednostka znajduje się w pobliskim otoczeniu, wykorzystywany jest kod przeznaczony dla jednostek bliskiego zasięgu, w przeciwnym przypadku wybierana jest logika jednostek zasięgowych.

### 5.9.2. Sztuczna inteligencja przyjaznych jednostek.

Kontrola jednostek przez gracza odbywa się za pomocą wyboru jednej z opcji w dwóch fazach:

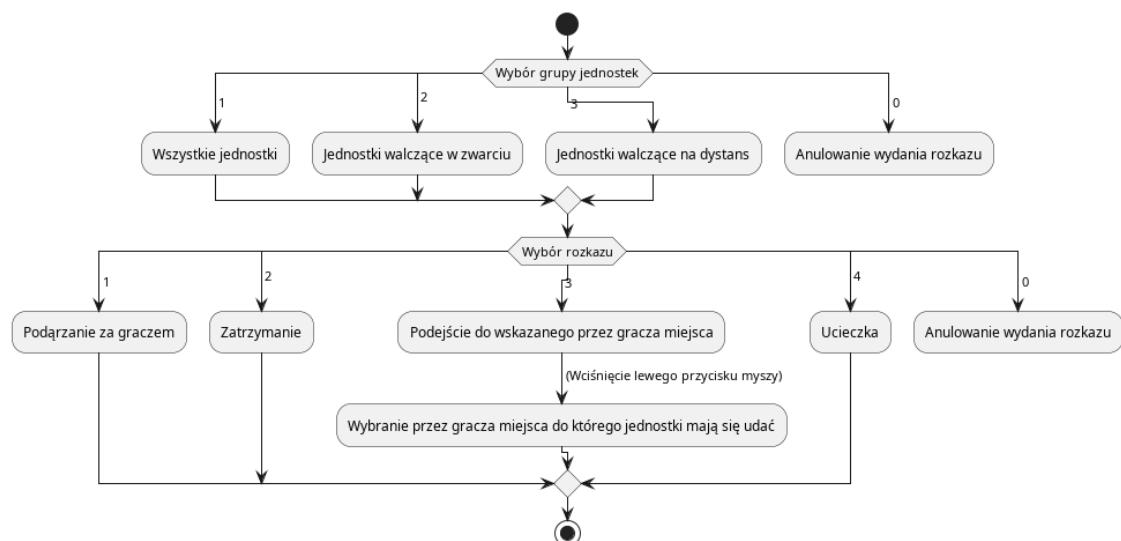
- Faza wybrania jednostek:

- wszyscy,
- jednostki bliskozasięgowe,
- jednostki hybrydowe,
- jednostki dalekozasięgowe,
- opcja anulowania wyboru.

- Faza wydania rozkazu:

- podążanie za graczem,
- zatrzymanie,
- atak,
- podejście do wskazanego przez gracza miejsca,
- ucieczka,
- opcja anulowania rozkazu.

Wydanie rozkazu polega na kliknięciu przycisku odpowiedzialnego za wejście w tryb wydawania poleceń, a następnie wybraniu numeru opcji reprezentującej grupę jednostek, której komenda ma dotyczyć. Ostatecznie należy podać numer rozkazu, który zostanie wydany. Przykładowa reprezentacja graficzna systemu jest widoczna na rysunku 2.8. Wykorzystanie tego modelu pozwala na łatwe rozwinięcie systemu o dodanie nowych metod wyboru grup jednostek oraz możliwych instrukcji dla przyjaznych agentów.



Rysunek 5.27: Wizualizacja przepływu sterowania podczas wydawania poleceń jednostkom.

## 5.10. Zapis i odczyt stanu gry (Bartosz Strzelecki)

Gracz w dowolnym momencie rozgrywki może przerwać grę i dokonać zapisu. Wykonuje to za pomocą skrótu klawiszowego F8. Powoduje to zapisanie na dysk danych wszystkich obiektów posiadających komponent Saveable. Strukturę serializowanych danych reprezentuje plik proto 5.4.

```
1 syntax = "proto3";
2
3 package base;
4
5 message Save {
6     repeated SaveableEntity entities = 1;
7 }
8
9 message Float3 {
10    float x = 1;
11    float y = 2;
12    float z = 3;
13 }
14
15 message Location {
16     Float3 position = 1;
17     Float3 rotation = 2;
18 }
19
20 message Param {
21     string key = 1;
22     string value = 2;
23 }
24
25 message SaveableEntity {
26     string name = 1;
27     Location transform = 2;
28     int32 hp = 3;
29     string prefab = 4;
30     repeated Param params = 5;
31 }
```

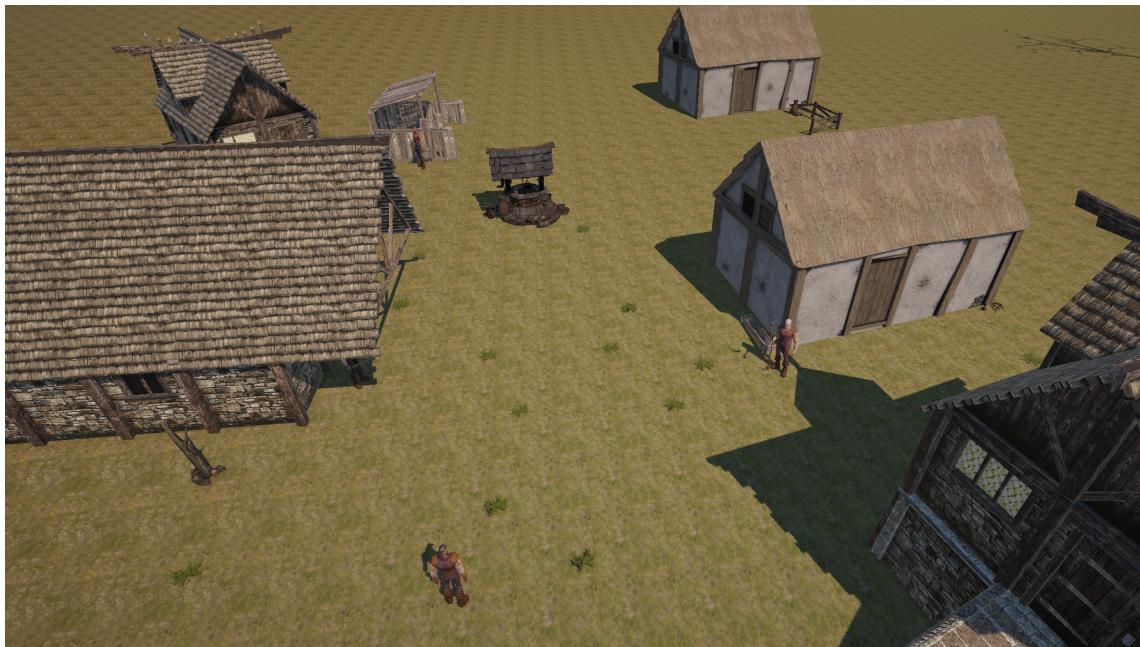
**Listing 5.4:** Plik .proto reprezentujący strukturę zapisu stanu gry

Powstały po zapisie plik jest przechowywany na dysku w lokalizacji zdefiniowanej przez stałą Environment.SpecialFolder.MyDocuments w formie binarnej. W tym przypadku zapamiętywane są jedynie dane, które zmieniają się w trakcie rozgrywki, takie jak położenie gracza, przeciwników, itd. Zawartość statyczna taka jak krzywizna terenu i flora występująca na mapie nie jest zapisywana, a jej wczytywanie jest rozwiązywane w trakcie ładowania sceny przez Unity Runtime.

W momencie, w którym gracz zażąda odczytania zapisu gry, następuje proces odwrotny. Odpowiednie obiekty zostają utworzone, a następnie przypisywane są im wartości na podstawie danych zawartych w pliku zapisu. Obiekty identyfikowane są po ich nazwie, w taki sposób, w jaki są zarządzane przez Unity. Tę akcję gracz wykonuję za pomocą przycisku F9 lub poprzez wybranie interesującego go zapisu z menu głównego.

### **5.11. Przebieg rozgrywki (Bartosz Strzelecki, Bogna Lew)**

Gracz zaraz po rozpoczęciu rozgrywki znajduje się w małej wiosce (rys. 5.28). Pierwszym elementem przyciągającym uwagę gracza, jest postać o imieniu Amargein (rys. 5.29), która oferuje nagrodę za pokonanie groźnego niedźwiedzia buszującego w pobliskim lesie. Użytkownik jednak szybko się orientuje, że nie jest w stanie pokonać bestii w pojedyńkę.



**Rysunek 5.28:** Kadr z gry przedstawiający wioskę, w której gracz rozpoczyna rozgrywkę.



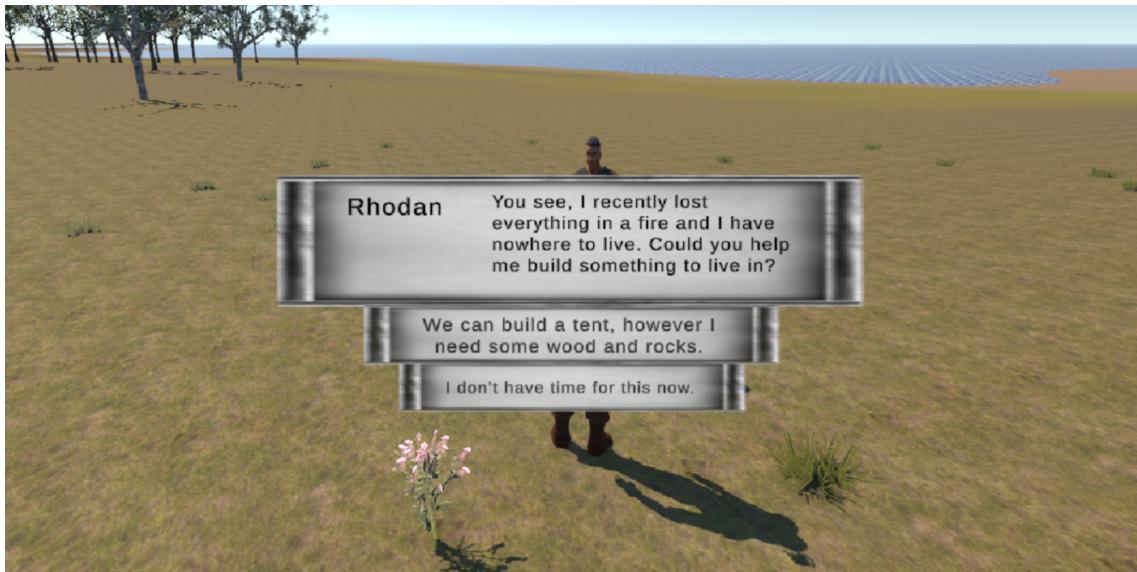
Rysunek 5.29: Kadr z gry przedstawiający postać niezależną zlecającą zadanie postaci gracza.

Po głębszych poszukiwaniach gracz znajduje dwójkę najemników (rys. 5.30), którzy byliby w stanie wesprzeć główną postać za odpowiednią opłatą. Początkowo użytkownik nie może sobie pozwolić na takie wydatki i jest zmuszony do znalezienia prostszego zadania w celu zadbania o odpowiedni stan finansów.



Rysunek 5.30: Kadr z gry przedstawiający możliwych do wynajęcia najemników.

Nieopodal wioski gracz dostrzega stojącego samotnie budowniczego. Po podejściu do niego, dowiaduje się, że nazywa się on Rhodan i potrzebuje pomocy. Z jego opowieści wynika, że niedawno stracił cały swój dobytek w pożarze i teraz nie ma gdzie mieszkać.



Rysunek 5.31: Kadr z dialogu z Rhodanem podczas którego opowiada swoją historię.

Gracz proponuje mu wybudowanie namiotu, jednakże wymaga to zasobów, których nie posiada. Zapytawszy Rhodana gdzie mógłby je zdobyć, dowiaduje się, że na zachodzie na skraju lasu leżą kłody oraz kamienie, które może spróbować zebrać. Zgodnie z sugestią, gracz postanawia udać się w to miejsce, aby uzbierać wymagane surowce.



Rysunek 5.32: Kadr z dialogu z Rhodanem z wytycznymi odnośnie zbierania zasobów.

Po zebraniu zasobów wraca porozmawiać z Rhodanem. Dowiaduje się od niego, że gdy mu wskaże odpowiednie do budowy namiotu miejsce, to postać podejdzie i go wybuduje. Tak jak został poinstruowany, gracz wybiera miejsce, które nadawałoby się pod budowę namiotu, a Rhodan podbiega i rozpoczyna budowę, wspomagając się młotkiem. Gdy mężczyzna skończył pracę gracz odbiera swoją nagrodę.



Rysunek 5.33: Kadr z gry, na którym widać budującego Rhodana.

Po wykonaniu uprzedniego zadania gracz jest już w stanie wykupić pomoc dwóch okolicznych na-jemników. Po rozmowie każdy z nich jest gotów zgodzić się na opłatę w wysokości 10 złotych monet. Tak przygotowana drużyna gracza jest gotowa na wyruszenie na przygodę, by uratować mieszkańców wioski od niedźwiedzia terroryzującego osadę.

Gracz, wykorzystując swoje zdolności nawigacji, udaje się na wschód i po krótkim czasie jest w stanie zidentyfikować zagrożenie. Widząc swój cel, gracz wydaje rozkaz ataku. Wynajęte jednostki szybko uporały się z niedźwiedziem. Każda jednostka wykorzystywała broń, z którą jest jak najbardziej biegła, czyli broń białą lub łuk.



Rysunek 5.34: Kadr z gry przedstawiający dwóch najemników toczących walkę z niedźwiedziem.

Po powrocie do wioski gracz jest uznany za bohatera, który ocalił wszystkich mieszkańców i zostaje odpowiednio nagrodzony.

## **6. PODSUMOWANIE**

Niniejszy rozdział zawiera podsumowanie osiągniętych rezultatów, ewentualne dalsze możliwości rozwoju oraz opinię o utworzonym prototype. W tym celu zespół poprosił osobę z zewnątrz o zagranie w przygotowaną grę oraz wyrażenie swojego zdania na jej temat.

### **6.1. Osiągnięte efekty (*Bogna Lew*)**

W ramach projektu został przygotowany prototyp gry strategii czasu rzeczywistego, posiadającej aspekty komputerowej gry fabularnej. Gra osadzona została w realiach wczesnego średniowiecza i czerpała inspirację z kultury celtyckiej. Została ona utworzona za pomocą silnika Unity oraz dostępnych dla niego narzędzi. Dodatkowo wykorzystano w niej modele i animacje, które są możliwe do pozyskania w Asset Store oraz stylistycznie oddają realia wybranej epoki historycznej.

Utworzony prototyp udostępnia podstawowe mechanizmy typowe dla gatunku gier strategii czasu rzeczywistego, takie jak zarządzanie jednostkami, budowanie budowli oraz prosty system zarządzania zasobami. Ponadto gra umożliwia użytkownikowi możliwość sterowania własną postacią oraz wchodzenia w interakcje z postaciami niezależnymi, od których może przyjmować zlecenia.

W grze występują cztery główne typy jednostek: postacie bez broni, miecznicy, łucznicy oraz budowniczowie. Pierwsze dwa rodzaje odpowiadają za walkę w zwarciu, wykonując odpowiednie ataki. Łucznicy preferują wykonywać ataki dystansowe, jednakże w razie konieczności podejmują walkę wręcz. Budownicze są jednostkami odpowiedzialnymi za budowanie budynków wskazanych przez gracza i nie wykonują żadnych ataków. Gracz może wydawać komendy postaciom wojowników, takie jak rozkaz ataku, podążania za jego awatarem bądź udania się we wskazane miejsce.

Mechanizm budowania udostępnia graczy cztery typy budowli: ławkę, namiot, dom oraz płot, a ich modele stylistycznie pasują do wybranej epoki historycznej. Gra stosuje typowe dla swojego gatunku ograniczenie, jakim jest konieczność poniesienia kosztów budowy obiektu. Ponadto gra uniemożliwia graczy umiejscowienie budowli na obszarze zbyt stromym lub na którym znajdują się inne obiekty bądź postacie.

Gra zawiera również prosty interfejs użytkownika, wspomagający gracza w trakcie rozgrywki. Oferuje między innymi kompas, na którym są zaznaczane najważniejsze punkty w świecie. Imituje to sposób nawigacji w epoce, w której osadzona jest gra, kiedy to ludzie nie posiadali pełnej wiedzy o otoczeniu, a jedynie pewne ogólne informacje.

Utworzona gra spełnia główne założenie projektu, jakim jest jak najdokładniejsze oddanie realiów historycznych wybranej epoki. Wykorzystywane w grze modele, bronie, czy słownictwo zostało starannie dobrane tak, aby dobrze oddawały czasy wczesnego średniowiecza i panujący wtedy światopogląd.

### **6.2. Opinia osoby z zewnątrz (*Zofia Sosińska*)**

Osoba niezależna została poproszona o zagranie w wytworzoną, prototypową grę. Następnie opisała swoje odczucia, które zostały przytoczone w tej pracy. O opinię poproszono osobę z wieloletnim doświadczeniem w graniu w różne rodzaje gier, aby miała ona odniesienie do produktów dostępnych na rynku.

Pierwszym odruchem po rozpoczęciu gry było założenie, że sterowanie postacią odbywa się za

pomocą klawiszy W, A, S i D oraz myszy. Spodziewano się, że sterowanie będzie wymagało tych komponentów, gdyż jest to najczęstsze rozwiązanie. Następnie logiczne wydało mu się użycie przycisku lewego klawisza Shift, aby przyspieszyć ruch postaci. Tak samo naturalne było użycie myszy do obrotu kamery, a całe sterowanie postacią zostało opisane jako przyjemne.

Po kolej odkrywano wszystkie mechaniki programu. Przyznano, że informacja o możliwej interakcji zwróciła uwagę na postaci, z którymi można było porozmawiać. Doceniono pieczętowe oddanie średniowiecznego stylu wypowiedzi w dialogach. Zabieg znacznie zwiększył immersję i umilił rozgrywkę. Zwrócono jednak uwagę, że dialogi są długie i przydatna byłaby mechanika wyświetlenia całego tekstu po naciśnięciu jakiegoś przycisku. Interfejs budowania wzbudził bardzo pozytywną reakcję. Uznano go za intuicyjny oraz przyjemny do użytku. Zwrócono uwagę na animację budowania budynku. Wykonana czynność podniosła poziom immersji gry. Przy nawigacji po świecie kompas odegrał kluczową rolę. Dzięki niemu to zadanie było proste. Szybko spostrzeżono, że lewym przyciskiem myszy można wyprowadzić atak, co było bardzo wygodne. Przy konfrontacji z niedźwiedziem dłuższą chwilę zajęło wyczucie odległości, z jakiej najlepiej go uderzyć i za lepszy pomysł uznano zatrudnienie najemników, którzy szybko poradzili sobie z bestią. Wydawanie im komend uznano za bardzo przydatną mechanikę, która wprowadziła element taktyczny. Jako miłe ułatwienie uznano dziennik ze streszczeniami zadań i podpowiedziami dotyczącymi mechanik gry.

Prototyp ogólnie uznano za dobrą bazę do późniejszego rozwijania. Projektanci zwracali uwagę na intuicyjność i wygodę sterowania, immersję gry i oddanie realiów wczesnego średniowiecza oraz zaimplementowali najpotrzebniejsze mechaniki gry RTS.

### **6.3. Dalsze możliwości rozwoju (Bartosz Strzelecki, Zofia Sosińska)**

Powyższy projekt jest jedynie prototypem zawierającym podstawowe mechaniki. Dalszy rozwój pracy głównie będzie polegać na konsolidacji zaimplementowanych systemów w gotowy produkt.

Głównym zadaniem byłoby opracowanie i realizacja rozwiniętej linii fabularnej, wykorzystując gotowe elementy wykonane w ramach projektu. Istotnym aspektem byłaby wymiana modeli i tekstur z tymczasowych, prototypowych zasobów, służących do celów demonstracyjnych zaimplementowanych mechanik na wyższej jakości i lepiej odzwierciedlające klimat rozgrywki. Gry nie różnią się w tym wypadku od innych produktów dostępnych na rynku. To, jak dobrze program wygląda tak samo wpłynie na pozytywny odbiór przez użytkownika, jak chociażby wykonanie i jakość materiałów, z których są wykonane elementy gier planszowych. "Estetyczne odczucia z grania w [...] gry ma znaczenie. Kiedy podnosisz pieczętowicie wyrzeźbiony pionek, reagujesz na niego doceniając aspekt estetyczny - jedną z form przyjemności." [20].

Kolejnym elementem byłoby skomponowanie muzyki dobrze oddającej atmosferę wczesnego średniowiecza oraz przygotowanie podkładów dźwiękowych, które zostałyby wykorzystane jako efekty dźwiękowe na przykład podczas wykonywanych animacji ataku, podnoszenia przedmiotów itd.

Dalszy plan rozwoju skupiałby się głównie na aspektach audiowizualnych. Powyższe aspekty pozytywnie wpłynęłyby na odbiór produktu przez potencjalnych przyszłych użytkowników.

## WYKAZ LITERATURY

- [1] G. Bobrek. "Historyczne bzdury, które na stałe trafiły do gier." (2020), [Online]. Available: <https://www.youtube.com/watch?v=-eySMANZ8Ok&t=2s>. (accessed: 24.06.2023).
- [2] A. Barnett, "Influencing players' perceptions of the past: Video game mechanics representing medieval marriages and betrothals," 2021. [Online]. Available: <https://knowledge-1uchicago-1edu-1000038z80025.han.bg.pg.edu.pl/record/3094>, (accessed: 25.06.2023).
- [3] A. Kramarzewski and E. D. Nucci, *Practical Game Design: Learn the Art of Game Design Through Applicable Skills and Cutting-Edge Insights*. Packt Publishing, 2018. [Online]. Available: <https://ebookcentral-1proquest-1com-13yl7b8z8001c.han.bg.pg.edu.pl/lib/pgpl/detail.action?docID=5353673>, (accessed: 02.07.2023).
- [4] D. Adams. "The state of the rts." (2006), [Online]. Available: <https://www.ign.com/articles/2006/04/08/the-state-of-the-rts>. (accessed: 17.05.2023).
- [5] B. Świątek. "15 najpopularniejszych strategii w historii." (2021), [Online]. Available: <https://www.gry-online.pl/S018.asp?ID=2903>. (accessed: 10.05.2023).
- [6] D. Pottinger. "Coordinated unit movement." (1999), [Online]. Available: <https://www.gamedeveloper.com/programming/coordinated-unit-movement>. (accessed: 26.07.2023).
- [7] R. Kremers, *Level Design: Concept, Theory, and Practice*. CRC Press LLC, 2009. [Online]. Available: <https://ebookcentral-1proquest-1com-13yl7b8z80008.han.bg.pg.edu.pl/lib/pgpl/detail.action?docID=1636255>, (accessed: 21.08.2023).
- [8] J. Argasiński et al., *Obrzym w cieniu: Gry wideo w kulturze audiowizualnej*. Wydawnictwo Uniwersytetu Jagiellońskiego, 2012.
- [9] S. Egenfeldt-Nielsen et al., *Understanding Video Games: The Essential Introduction*. Taylor & Francis Group, 2012. [Online]. Available: <https://ebookcentral-1proquest-1com-13yl7b87n0076.han.bg.pg.edu.pl/lib/pgpl/detail.action?docID=1181119>, (accessed: 29.07.2023).
- [10] E. Adams, *Projektowanie gier. Podstawy. Wydanie II*. Paraglyph, Incorporated, 2010. [Online]. Available: <https://helion.pl/pobierz-fragment/prgpo2/pdf>, (accessed: 10.08.2023).
- [11] R. M. (ed.), *User Interfaces*. InTech, 2010. [Online]. Available: [https://mts.intechopen.com/storage/books/6115/authors\\_book/authors\\_book.pdf](https://mts.intechopen.com/storage/books/6115/authors_book/authors_book.pdf), (accessed: 21.08.2023).
- [12] J. Spolsky, *User Interface Design for Programmers*. Apress, 2001. [Online]. Available: <https://link-springer-1com-100005dz80064.han.bg.pg.edu.pl/book/10.1007/978-1-4302-0857-0>, (accessed: 12.07.2023).
- [13] B. Ellison. "Defining dialogue systems." (2008), [Online]. Available: <https://www.gamedeveloper.com/design/defining-dialogue-systems>. (accessed: 27.10.2023).
- [14] C. Harteveld, *Triadic Game Design*. Springer, 2011. [Online]. Available: <https://link-springer-1com-100005dz8005b.han.bg.pg.edu.pl/book/10.1007/978-1-84996-157-8>, (accessed: 12.08.2023).
- [15] B. L. Mitchell, *Game Design Essentials*. John Wiley & Sons, Incorporated, 2012. [Online]. Available: <https://ebookcentral-1proquest-1com-13yl7b87n0076.han.bg.pg.edu.pl/lib/pgpl/detail.action?docID=818112>, (accessed: 11.07.2023).
- [16] U. Technologies. "Unity manual." (2023), [Online]. Available: <https://docs.unity3d.com/Manual/>. (accessed: 13.08.2023).
- [17] Google. "Protocol buffers documentation." (2023), [Online]. Available: <https://protobuf.dev>. (accessed: 14.08.2023).

- [18] J. G. Bond, *Projektowanie gier przy użyciu środowiska Unity i języka C#*. Od pomysłu do gotowej gry. Wydanie II. Helion, 2019.
- [19] K. Lammers, *Unity Shaders and Effects Cookbook*. Packt Publishing, 2013.
- [20] R. Koster, *Theory of Fun for Game Design*. Paraglyph, Incorporated, 2004. [Online]. Available: <https://ebookcentral-1proquest-1com-13yl7b8z80057.han.bg.pg.edu.pl/lib/pgpl/reader.action?docID=3384727>, (accessed: 05.07.2023).

## WYKAZ RYSUNKÓW

1.1	Mapa basenu Morza Śródziemnego z czasów Imperium Rzymskiego . . . . .	7
2.1	Budowanie budynku przez dedykowaną do tego jednostkę w grze <i>Warhammer 40,000: Dawn of War</i> . . . . .	14
2.2	Przykładowa postać przeciwnika utworzona przez system Nemesis. . . . .	15
2.3	Lewa część paska z informacjami w grze <i>Warcraft 3</i> . . . . .	16
2.4	Prawa część paska z informacjami w grze <i>Warcraft 3</i> . . . . .	16
2.5	Wyświetlenie dostępnych pułapek w grze <i>Orcs must die!</i> . . . . .	17
2.6	Przykład koła dialogowego w grze <i>Mass Effect</i> . . . . .	18
2.7	Kadr z gry <i>Fallout 3</i> przedstawiający przykładowy dialog. . . . .	19
2.8	Wykaz dostępnych rozkazów z gry <i>Mount&amp;Blade</i> . . . . .	20
2.9	Kompas z gry <i>Skyrim</i> . . . . .	20
2.10	Ekran rozwoju postaci z gry <i>TESV: Skyrim</i> . . . . .	22
2.11	Ekran rozwoju postaci z gry <i>Wiedźmin 3: Dziki Gon</i> . . . . .	23
2.12	Przykładowe elementy widzenia przez horyzont w grze <i>Dead by Daylight</i> . . . . .	24
2.13	Przykładowy efekt aury w grze <i>Dead by Daylight</i> . . . . .	24
2.14	System celowania występujący w grze <i>Phoenix Point</i> . . . . .	25
3.1	Przykładowa mapa wysokości (po lewej) oraz wygenerowany na jej podstawie teren (po prawej). . . . .	27
3.2	Widok na teren z drzewami. . . . .	27
3.3	System nawigacji w Unity. . . . .	28
3.4	Przykład zasobów typu ScriptableObject. . . . .	29
4.1	Harmonogram przedstawiony w postaci diagramu gantt. . . . .	30
4.2	Diagram przypadków głównych mechanik gry. . . . .	32
4.3	Diagram stanów gry. . . . .	33
4.4	Diagram klas gry. . . . .	34
4.5	Projekt interfejsu podstawowego UI. . . . .	36
4.6	Projekt menu stawiania budynków. . . . .	36
4.7	Projekt menu wydawania komend. . . . .	37
4.8	Projekt menu zapisu. . . . .	37
4.9	Projekt grafiki informującej o możliwym rozpoczęciu rozmowy. . . . .	38
4.10	Projekt dziennika z aktualnie zaczętym i nieukończonym zadaniem. . . . .	39
4.11	Projekt ekranu końca gry. . . . .	39
4.12	Projekt menu głównego. . . . .	40
5.1	Implementacja paska z najważniejszymi informacjami o stanie gry: aktualnym czasie, posiadanych surowcach i funduszach, położeniu i otoczeniu gracza, zdrowiu postaci oraz ikona sygnalizująca, czy użytkownik ma do wykonania jakieś zadanie. . . . .	44

5.2	Implementacja menu stawiania budynków, na którym pokazane są możliwe do zbudowania budowle i szczegółowe informacje o ich dostępności. . . . .	45
5.3	Implementacja menu wydawania komend. . . . .	46
5.4	Implementacja menu zapisu. . . . .	46
5.5	Implementacja grafiki informującej o możliwości rozpoczęcia rozmowy. . . . .	47
5.6	Implementacja grafiki informującej o możliwości podniesienia przedmiotu. . . . .	47
5.7	Implementacja dziennika z aktualnie zaczętym i nieukończonym zadaniem. . . . .	48
5.8	Implementacja menu zapisu. . . . .	49
5.9	Implementacja menu głównego. . . . .	50
5.10	Rozpiska elementów: a. główny pasek, b. symbol środka, c1., c2. paski końców kompasu.	50
5.11	Wizualizacja przypadku, w którym gracz patrzy centralnie na obozowisko wrogów. Na przeciwnieko oraz po lewej stronie znajdują się przeciwnicy, co jest zasygnalizowane na kompasie za pomocą symboli mieczy. Znajduje się na nim także informacja, że bohater jest lekko odchylony od Zachodu. . . . .	51
5.12	Rozmieszczenie zasymulowanych stron świata. . . . .	52
5.13	Widok po przesunięciu kamery w górę. . . . .	53
5.14	Widok po przesunięciu kamery w dół. . . . .	54
5.15	Komunikat informujący gracza o zdobyciu kolejnego poziomu umiejętności. . . . .	54
5.16	Wykres przedstawiający funkcję opisującą natężenie efektu w animacji markera. . . . .	55
5.17	Zachowanie programu cieniącego w przypadku przysłaniania markera przez przeszkody.	56
5.18	Przykładowa konwersacja z wykorzystaniem Dialogue Editor. . . . .	57
5.19	Przykładowe okno dialogowe widziane z perspektywy gracza. . . . .	57
5.20	Przykład podglądu w poprawnym umiejscowieniu. . . . .	58
5.21	Przykład podglądu w niepoprawnym umiejscowieniu. . . . .	59
5.22	Widok z góry na siatkę promieni służących do wykrywania kolizji z drzewami. . . . .	59
5.23	Widok z boku na siatkę promieni służących do wykrywania kolizji z drzewami. . . . .	59
5.24	Obraz przedstawia zasięgi odpowiednich regionów. . . . .	60
5.25	Diagram przedstawiający przepływ stanów dla sztucznej inteligencji przeciwników. . . . .	61
5.26	Reprezentacja graficzna celowania widziana z perspektywy jednostki dalekozasięgowej. Czerwony okrąg reprezentuje obszar, w którym znajdzie się 50% pocisków. Żółty obszar pokazuje maksymalny rozrzut. . . . .	61
5.27	Wizualizacja przepływu sterowania podczas wydawania poleceń jednostkom. . . . .	62
5.28	Kadr z gry przedstawiający wioskę, w której gracz rozpoczyna rozgrywkę. . . . .	64
5.29	Kadr z gry przedstawiający postać niezależną zlecającą zadanie postaci gracza. . . . .	65
5.30	Kadr z gry przedstawiający możliwych do wynajęcia najemników. . . . .	65
5.31	Kadr z dialogu z Rhodanem podczas którego opowiada swoją historię. . . . .	66
5.32	Kadr z dialogu z Rhodanem z wytycznymi odnośnie zbierania zasobów. . . . .	66
5.33	Kadr z gry, na którym widać budującego Rhodana. . . . .	67
5.34	Kadr z gry przedstawiający dwóch najemników toczących walkę z niedźwiedziem. . . . .	68

## **WYKAZ TABEL**

2.1 Porównanie gatunków gier.	12
3.1 Porównanie silników.	26
4.1 Rozwijalne umiejętności gracza.	41

## **ZAŁĄCZNIK A - LISTA WYKORZYSTANYCH KOMPONENTÓW**

### **Unity**

<https://unity.com>  
UNITY SOFTWARE LICENSE AGREEMENT 4.X

### **Protocol Buffers**

<https://github.com/protocolbuffers/protobuf>  
BSD-3-Clause license

### **System.Runtime.CompilerServices.Unsafe**

<https://www.nuget.org/packages/System.Runtime.CompilerServices.Unsafe/>  
MIT License

### **Blacksmith**

<https://assetstore.unity.com/packages/3d/environments/fantasy/blacksmith-46265>  
Standard Unity Asset Store EULA

### **DETAILED - Medieval Wells & Props**

<https://assetstore.unity.com/packages/3d/environments/fantasy/detailed-medieval-wells-props-177037>  
Standard Unity Asset Store EULA

### **Awesome Free Scans - Wood Logs**

<https://assetstore.unity.com/packages/3d/props/exterior/awesome-free-scans-wood-logs-216756>  
Standard Unity Asset Store EULA

### **Pixel Art Icon Pack - RPG**

<https://assetstore.unity.com/packages/2d/gui/icons/pixel-art-icon-pack-rpg-158343>  
Standard Unity Asset Store EULA

### **Dialogue Editor**

<https://assetstore.unity.com/packages/tools/utilities/dialogue-editor-168329>  
Standard Unity Asset Store EULA

### **Gardening Tools Pack - 26 PBR objects**

<https://assetstore.unity.com/packages/3d/props/tools/gardening-tools-pack-26-pbr-objects-183477>  
Standard Unity Asset Store EULA

### **3D The Blacksmith's House**

<https://assetstore.unity.com/packages/3d/environments/fantasy/3d-the-blacksmith-s-house-252972>  
Standard Unity Asset Store EULA

### **White Rabbit**

<https://assetstore.unity.com/packages/3d/characters/animals/white-rabbit-138709>  
Standard Unity Asset Store EULA

### **Rock package**

<https://assetstore.unity.com/packages/3d/props/exterior/rock-package-118182>  
Standard Unity Asset Store EULA

**FREE - Modular Character - Fantasy RPG Human Male**

[https://assetstore.unity.com/packages/3d/characters/humanoids/humans/...](https://assetstore.unity.com/packages/3d/characters/humanoids/humans/)  
Standard Unity Asset Store EULA

**FREE Stylized Bear - RPG Forest Animal**

<https://assetstore.unity.com/packages/3d/characters/animals/free-stylized-bear-rpg-forest-animal-228910>  
Standard Unity Asset Store EULA

**Free Ui Pack**

<https://assetstore.unity.com/packages/2d/gui/icons/free-ui-pack-170878> Standard Unity Asset Store EULA

**PBR medieval houses pack**

<https://assetstore.unity.com/packages/3d/environments/fantasy/pbr-medieval-houses-pack-71546>  
Standard Unity Asset Store EULA

**LOWPOLY - Weapon Pack**

<https://assetstore.unity.com/packages/3d/props/weapons/lowpoly-weapon-pack-216486> Standard Unity Asset Store EULA