Politechnika Wrocławska

Wydział Informatyki i Zarządzania

Informatyka
specjalność: brak

Praca dyplomowa – inżynierska

Gra survival wykorzystująca technologie wirtuanlej rzeczywistości przy użyciu silnika Unity.

Bartosz Matuszczyk

słowa kluczowe:
VR
Wirtualna rzeczywistość
Gra survival

krótkie streszczenie:
Praca zawiera projekt oraz implementacje gry survival z wykorzystaniem technologii VR przy użyciu silnika Unity.

opiekun pracy dyplomowej	.....	.....	.....
	Tytuł/stopień naukowy/imię i nazwisko	*ocena*	*podpis*

Ostateczna ocena za pracę dyplomową

Przewodniczący Komisji egzaminu dyplomowego	.....	.....	.....
	Tytuł/stopień naukowy/imię i nazwisko	*ocena*	*podpis*

*Do celów archiwalnych pracę dyplomową zakwalifikowano do:**
a) kategorii A (akta wieczyste)
b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)
** niepotrzebne skreślić*

pieczątka wydziałowa

Wrocław

2019

Streszczenie

W niniejszej pracy zawarty został opis projektu oraz implementacji gry survival w technologii wirtualnej rzeczywistości przy użyciu silnika Unity. Głównym zakresem pracy jest zaprojektowanie oraz implementacja prototypu gry zgodnie z przyjętym procesem wytwarzania oprogramowania.

Praca składa się z siedmiu części. Pierwsza to wstęp do problematyki pracy, krótko przedstawia czym jest wirtualna rzeczywistość. W drugiej części określone zostały wymagania projektowe, w postaci spisanych wymagań klienta. Trzeci rozdział traktuje o technologiach i technikach, które są przydatne do implementacji prototypu, oraz przedstawia istniejące podobne rozwiązania. W czwartym rozdziale zaprezentowano założenia projektowe dotyczące prototypu, które powstały na bazie analizy rozdziału drugiego. Piąty rozdział zawiera najważniejsze informacje o projekcje takie jak historyjki użytkownika czy opis interfejsu, ale również schemat implementacji. Szósta część opisuje schematy działania oprogramowania oraz sposób testowania. Ostatni rozdział poświęcony jest podsumowaniu całej pracy, w którym można znaleźć informacje o grze, takie jak wygląd, informacje o instalacji i krótkie wnioski

Abstract

This thesis consists description of design and implementation of survival game with virtual reality technology in Unity Engine. The main scope of this thesis is to design and implement a game prototype in according to the adopted software development process.

This work consists of seven parts. The first is a preface of problem of work, briefly presents what virtual reality is. In the second part, design requirements are defined, as user requirements. The third chapter is about technologies and techniques that are useful for the implementation of the prototype, it also presents similar existing solutions. The fourth chapter presents the design assumptions concerning the prototype. They were based on the analysis of the second chapter. The fifth chapter contains the most important information about project, such as user stories or interface description, but also the implementation project. The sixth part is description of high level architecture. The last chapter is devoted to summary of the whole thesis, you can find information about the game, such as its overview, information about installation and short conclusion

Spis treści

1. Wstęp

1.1. Wprowadzenie do problematyki

Wirtualna rzeczywistość od początku jej istnienia była ciekawym zagadnieniem[1]. Technologia ta wykorzystuje do działania zmysły człowieka, głównie wzrok, słuch i dotyk. Poprzez specjalne gogle oraz wykorzystanie dwóch ekranów lub w przypadku smartphona jest to jeden ekran, który wyświetla dwa obrazy przesunięte nieco względem siebie możliwe jest stworzenie imitacji trójwymiarowego świata. Sterowanie jest możliwe dzięki specjalnym kontrolerom lub zwykłemu kontrolerowi do konsoli.

Programy korzystające z wirtualnej rzeczywistości, starają się jak najlepiej oddać wirtualny świat. Niestety aplikacje działające na urządzeniu mobilnym, z powodów niewystarczającej mocy obliczeniowej, są zmuszone do ukazywania świata gorszej jakości niż w przypadku okularów takich jak Oculus Rift[2].

W niniejszej pracy implementowana będzie gra survival. Jest to gra polegająca na przetrwaniu jak najdłuższy czas dostaniu się w odpowiednie miejsce. Aby utrudnić to zadanie graczowi, na mapie znajdują się przeciwnicy oraz zmniejszające się z czasem paski statusów, które po spadnięciu do zera aktywują negatywne efekty. Gracz ma oczywiście możliwość uzupełniania pasków poprzez różne przedmioty występujące na mapie.

1.2. Geneza pracy

Rynek gier, który wykorzystuje wirtualną rzeczywistość jest bardzo ubogi. W szczególności brakuje gier, gdzie mechanika jest skomplikowana w znacznym stopniu. Z reguły są to gry proste niewymagające wiele wysiłku od użytkownika, , jak na przykład:

- BAMF VR
- Need for Jump (VR game)
- Flats
- InMind VR (Cardboard)

Społeczność graczy, którzy korzystają z gier wirtualnej rzeczywistości nie jest zbyt duża. Odpowiada za to głównie cena sprzętu, dlatego też nie ma wielu osób będących w stanie wypełnić lukę, jaką jest brak dostępności ciekawych gier z obszaru wirtualnej rzeczywistości.

1.3. Cel pracy

Celem pracy jest wykonanie projektu gry survival wykorzystującej wirtualną rzeczywistość w środowisku Unity[3].

1.4. Zakres pracy

Rozdział 1. jest rozdziałem wstępnym, który wprowadza w problematykę pracy (1.1.), jej genezę (1.2.), cel (1.3.), oraz zakres (1.4.), natomiast w podrozdziale 1.5. zawarte zostaną podziękowania.

Rozdział 2. jest poświęcony wprowadzeniu do stanu wiedzy oraz techniki w zakresie tematyki pracy (2.1.), przeglądzie istniejących rozwiązań (2.2.) oraz technik i technologii (3.3.).

Rozdział 3. przedstawaia wymagania projektowe (4.1), sposób realizacji pracy, jej przedmiot (4.2.), wymagania funkcjonalne (4.3.) oraz niefunkcjonalne (4.4.) jak i opis podstawowej architektury systemu (4.5.)

Rozdział 5. prezentuje prowadzenie do projektu gry (5.1.), jej przypadki użycia (5.2.), interfejs (5.3.) oraz ogólną architekturę projektu (5.4.)

Rozdział 6. Składa się z krótkiego wprowadzenia do implementacji(6.1), jakie zostały wykorzystane środowiska oraz narzędzia (6.2), opis architektury wysokiego poziomu (6.3), informacje dotyczące instalacji oprogramowania (6.4), testów gry (6.5) oraz podsumowania (6.6).

Rozdział 7. poświęcony został na podsumowanie całej pracy dyplomowej. Składa się z opisu wykonanych prac (7.1), czy cel pracy został zrealizowany (7.2), wniosków (7.3) oraz perspektyw dalszego rozwoju (7.4).

1.5. Podziękowania

Dziękuję promotorowi Maciejowi Drwalowi za poświęcony mi czas oraz cenne rady.

Dziękuję rodzicom Beacie i Witoldowi za wspieranie mnie podczas trudnych miesięcy, poświęconych na pisanie pracy dyplomowej.

Szczególnie dziękuję mojej siostrze Martynie za wsparcie oraz pomoc, jakiej mi udzieliła.

2. Stan wiedzy i techniki w zakresie tematyki pracy

2.1. Wprowadzenie

Rozdział ten zawierać będzie przegląd podobnych rozwiązań z tematyki pracy inżynierskiej. W tym przypadku będą to inne gry, te wykorzystujące technologie wirtualnej rzeczywistości oraz te, które nie wykorzystują tej technologii. Dodatkowo zostaną wypisane ich podobieństwa oraz różnice. W tym rozdziale znajdzie się również opis technik, które zostaną wykorzystane do implementacji prototypu, jak również krótki opis technologii.

2.2. Przegląd podobnych istniejących rozwiązań

W rozdziale tym zostaną przedstawione oraz porównane rozwiązania, które są podobne do realizowanego prototypu. Niestety, nie ma wiele podobnych rozwiązań, które wykorzystują wirtualną rzeczywistość w sposób zrealizowany w pracy inżynierskiej. Po opisie każdego rozwiązania znajdować się będzie lista podobieństw oraz różnic między daną grą a prototypem.

Pierwsza gra, która zostanie opisana, to *Unturned[4]*. Jest to gra komputerowa z gatunku survival horror z elementami sandboxu i widokiem pierwszoosobowym z możliwością zmiany na widok trzecioosobowy. Rozgrywa się w otwartym postapokaliptycznym świecie opanowanym przez zombie. Grafika tej gry stylizowana jest na podstawie kultowej gry *Minecraft*, czyli proste, a raczej „kwadratowe" modele 3D. W grze jest bardzo bogaty wybór broni oraz przedmiotów. Istnieje możliwość tworzenia ogromnych fortyfikacji, elektroniczne zabezpieczenia oraz zmyślne pułapki. Gra zapewnia również użytkownikowi dostęp do rozmaitych pojazdów, dzięki której łatwiej poruszać się po ogromnym świecie. Posiada dwa tryby rozgrywki, jednoosobowy oraz wieloosobowy. Ponadto można wybrać z dwóch rodzajów rozgrywki. W trybie areny, gdzie gracze walczą między sobą na cyklicznie ograniczającym się obszarze, podobnie jak w grach *Battle Royal*, lub w trybie przetrwania, gdzie trafia się do otwartego świata. Celem użytkownika jest walka o przetrwanie z hordami zombie. W tym trybie gracze mogą sobie pomagać, ale również ze sobą walczyć.

- Podobieństwa:
  - Grafika
  - Modele
  - Tryb survival
- Różnice:
  - Wirtualna rzeczywistość
  - Interfejs
  - Multiplayer
  - Więcej współczynników
  - Otwarty świat
  - Tryby gry

Kolejnym rozwiązaniem jest gra *SuperHo[5]t*. Jest to gra komputerowa z gatunku *First Person Shooter*. Rozgrywka toczy się w zamkniętych lokacjach, które następują po sobie. Celem gry jest wyeliminowanie przeciwników w każdej lokacji. Gracz dostaje do dyspozycji kilka różnych rodzai nie tylko broni palnej, ale również broni miotanej, takiej jak noże do rzucania czy shurikeny. Dodatkowo może wykorzystać przedmioty znajdujące się w danej lokacji, m.in. popielniczkę czy kubka. Do zabicia przeciwników wystarczy jeden celny strzał, rzut lub uderzenie, ale gracz ma podobnie ograniczoną wytrzymałość. Dość oryginalną mechaniką tej gry jest specyficzny upływ czasu, to znaczy, że czas biegnie tylko i wyłącznie, gdy gracz się porusza. Styl graficzny gry jest stworzony na zasadzie kontrastu,

głównymi kolorami lokacji są odcienie bieli, przeciwnicy mienią się na czerwono natomiast broń jest czarna, przez co wszystko jest bardzo widoczne.

- Podobieństwa:
  - Eliminacja wszystkich przeciwników
  - Wirtualna rzeczywistość
  - Ograniczony świat
- Różnice:
  - Grafika
  - Czas biegnie tylko podczas poruszania się
  - Wytrzymałość gracza oraz przeciwników
  - Zachowanie przeciwników

Następnymi rozwiązaniami, a raczej grupą rozwiązań są gry z gatunku *Battle Royal,* takie jak *Player's Unknown Battleground[6]* czy *Apex Legend[7]s.* Celem gier z gatunku *Battle Royal* jest eliminacja wszystkich wrogich graczy oraz przetrwanie dłużej niż inni. Koniec gry następuje, gdy przy życiu pozostanie tylko jeden gracz. Akcja toczy się na zamkniętej, aczkolwiek dużej mapie z wieloma zróżnicowanymi lokacjami, gdzie występują różne rodzaje przedmiotów. Należą do nich kamizelki, hełmy oraz plecaki, które posiadają trzy poziomy, im wyższy poziom, tym lepszy dany atrybut. Występują również przedmioty konsumpcyjne, które umożliwiają wyleczenie gracza oraz zwiększają prędkość poruszania się. Gracze mogą też znaleźć wiele rodzajów uzbrojenia. Od broni krótkiej przez pistolety maszynowe aż po karabiny, a dodatkowo ich ulepszenia, takie jak różnego rodzaju magazynki, dodatkowa optyka oraz tłumiki i uchwyty. Przedmioty te rozmieszane są na mapie w sposób losowy we wcześniej przygotowanych punktach, ułatwiają one znacząco przetrwanie w tym świecie. Wraz z upływem czasu zmniejsza się liczba graczy, co jest rekompensowane przez cyklicznie ograniczany obszar gry, aby zwiększyć prawdopodobieństwo spotkania się dwóch graczy. Gra zapewnia również różne pojazdy, dzięki którym można szybciej przemieszczać się po mapie.

- Podobieństwa:
  - Zbieranie przedmiotów
  - Rozmieszczanie przedmiotów
- Różnice:
  - Brak wirtualnej rzeczywistości
  - Rozgrywka w trybie gracz vs gracz
  - Zawężanie sie terenu rozgrywki wraz z czasem gry.

Uncrowded:

Ostatnim przedstawionym rozwiązaniem jest *Uncrowded[8],* bardzo podobna do *Unturned,* gra komputerowa z gatunku survival horror z otwartym światem. Celem jest przetrwanie w świecie opanowanym przez zombie. Jedynym możliwym trybem jest survival. Przy rozpoczęciu gry, gracz wybiera jedną z pięciu dostępnych postaci: policjanta, farmera, sprzedawcę lub doktora, różnią się one początkowym ekwipunkiem. Gracz dostaje do dyspozycji bronie palne oraz białe, rozmieszczone w sposób losowy w świecie gry. Istnieje kilka różnych rodzajów przeciwników, od powolnych słabych zombie, które bardzo łatwo jest zabić, do szybkich i wytrzymałych, którzy bardzo szybko skracają dystans do gracza, aby móc go zaatakować. Podobnie jak w przypadku *Unturned* gra daje możliwość wykorzystania pojazdów oraz tworzenia budowli. Do rozgrywki dodano również bardzo prosty system tworzenia przedmiotów.

- Podobieństwa:

Z komentarzem [1]: Nie mogłem znaleźć dobrego opisu, więc skorzystałem z youtube, nie jestem pewien czy mogę to dać jako link do bibliografii.

04.03.2019

    - Podobnie jak *Unturned*
    - Parametry głodu i pragnienia
    - Różne typy przeciwników
- Różnice:
    - Wirtualna rzeczywistości
    - Interfejs
    - Mapa
    - System RPG

2.3. Przegląd technik i technologii

Niniejszy rozdział przedstawia różne technologie, które zostaną wykorzystane do implementacji prototypu oraz wykorzystania prototypu w praktyce.

System operacyjny Android[9], stworzony przez firmę Google, jest to jeden z czterech najbardziej popularnych systemów operacyjnych dla telefonów komórkowych. Bazuje on na zmodyfikowanej wersji Linux'a. System ten dostosowany jest głównie do urządzeń dotykowych. Prototyp będzie działał na urządzeniach, które posiadają system operacyjny Android.

Język C# jest językiem obiektowym stworzonym przez Andreas Hejlsberg'a dla firmy Microsoft[10]. Program napisany w C# kompilowany jest do Common Intermediate Language. Jest to kod pośredni wykonywany w środowisku takim jak .NET Framework. Jeśli system operacyjny nie posiada tego środowiska, nie jest możliwe korzystanie z tego języka. Wykorzystany zostanie on do pisania skryptów w Unity.

Bluetooth[11] jest to technologia łączności bezprzewodowej wykorzystująca fale radiowe o częstotliwości 2.4GHz. Umożliwia ona komunikację między dwoma urządzeniami wykorzystującymi tą technologię, która zaprojektowana została przez firmę Ericsson i pojawiła się na rynku w 1999 roku. Wykorzystana będzie do komunikacji kontrolera Xbox z urządzeniem mobilnym.

Unity jest to silnik gry stworzony przez firmę Unity Technologies w 2005 roku.[12] Silnik może zostać wykorzystany do tworzenia gier dwuwymiarowych, trójwymiarowych, poszerzonej rzeczywistości oraz wirtualnej rzeczywistości. W tym przypadku zostanie wykorzystany do stworzenia prototypu korzystającego z wirtualnej rzeczywistości.

Wirtualna rzeczywistość jest to technologia wykorzystująca technologię informatyczną, aby stworzyć w pełni trójwymiarowy obraz świata lub przedmiotu. Wykorzystywana jest ona do tworzenia realnych symulacji oraz gier. Prototyp będzie wykonany w tej technologii.

2.4. Podsumowanie

Na rynku nie ma wielu gier wykorzystujących wirtualną rzeczywistość. Można spotkać głównie proste symulacje wykorzystujące ją, takie jak symulator kolejki górskiej, czy proste gry polegające na chodzeniu przez pomieszczenia i wzbudzaniu w graczu strachu. Projekt ten może zostać ciepło przyjęty przez niezbyt liczną społeczność użytkowników wirtualnej rzeczywistości.

3. Założenia projektowe

3.1. Wymagania projektowe

Aplikacja tworzona jest w celu umieszczenia na platformach sprzedających lub udostępniających gry mobilne, takich jak Google Play czy App Store. W aplikacjach wykorzystujących wirtualną rzeczywistą, wyświetlanie reklam może być nieprzyjemnym doświadczeniem, zmniejszającym komfort grania, dlatego preferowana jest sprzedaż aplikacji za pewną cenę.

Główną funkcją aplikacji jest gra w wirtualnej rzeczywistości umożliwiająca:

- Poruszanie się gracza
- Podnoszenie broni oraz przedmiotów przez gracza
- Atakowanie przeciwników
- Otrzymywanie obrażeń

Ponadto, dla utrudnienia rozgrywki, gracz musi zarządzać swoimi zasobami, zbierać je oraz wykorzystywać w odpowiednim czasie, aby gra nie była monotonna. Przeciwnicy muszą się od siebie różnić pod pewnymi względami takimi jak:

- Ilość wytrzymałości
- Prędkość poruszania się
- Siła zadanych obrażeń

Przeciwnicy powinny dzielić się na dwie grupy:

- Poruszające się losowo
- Broniące pewnego miejsca

Do działania aplikacji wymagane jest:

- System Android
- Specjalne okulary obsługujące wirtualną rzeczywistość
- Kontroler Xbox

W późniejszym czasie możliwe będzie przeniesienie aplikacji na urządzenia z systemem IOS lub inne dowolne urządzenie, które jest w stanie obsłużyć wirtualną rzeczywistość.

Aplikacja jest grą jednoosobową, więc wykorzystywana może być tylko przez jednego użytkownika. Gracze tej aplikacji to głównie osoby zafascynowane wirtualną rzeczywistością, które potrafią obsługiwać smartphona oraz kontroler do gier. Aplikacja uruchamia się poprzez wybranie jej ikony z menu smartphona. W przyszłości można dodać do gry tryb sieciowy, co zwiększy grywalność.

3.2. Przedmiot prac

Przedmiotem projektu jest wykonanie gry typu survival z obsługą technologii wirtualnej rzeczywistości na urządzenia mobilne z systemem Android. Gra polegać będzie na przetrwaniu w labiryncie oraz oczyszczeniu go z przeciwników. Przetrwanie gracza będzie zależeć od trzech parametrów:

- Życia

- Głodu
- Pragnienia

Parametry przedstawione zostaną w postaci pasków w trzech kolorach. Czerwony dla życia, niebieski dla pragnienia oraz żółty dla głodu. Jeśli pasek życia spadnie do zera, gra zakończy się przegraną. W przypadku, gdy wartości pasków głodu i pragnienia spadną do zera, gracz otrzymywać będzie odpowiednią karę do prędkości poruszania się oraz obrażenia w czasie.

Gracz do dyspozycji będzie mieć różne rodzaje broni oraz przedmioty redukujące pragnienie, głód oraz odnawiające życie, a także plecaki, dzięki którym będzie możliwe przechowywanie większej ilości przedmiotów.

Przedmioty rozmieszczane są na mapie w losowych miejscach.

Na drodze gracza do zwycięstwa staną przeciwnicy, którzy różnią się parametrami takimi jak:

- Życie
- Prędkość poruszania się
- Rozmiar
- Zadawane obrażenia

Będą one patrolować labirynt w poszukiwaniu gracza, a gdy go zauważą lub usłyszą podążą za nim i zaczną go atakować.

3.3. Wymagania funkcjonalne

W niniejszym rozdziale przedstawione zostaną wymagania funkcjonalne, które muszą zostać zaimplementowane w dostarczonym prototypie.

1. Gracz obraca się przy pomocy ruchów głową.
2. Gracz ma możliwość poruszania się w dowolnym kierunku.
3. Gracz ma możliwość skakania.
4. Gdy gracz skacze, zwiększony jest zasięg jego wykrycia.
5. Gracz ma możliwość skradania się.
6. Gdy gracz się skrada, zmniejszony jest zasięg jego wykrycia.
7. Z upływem czasu zmniejszają się wskaźniki głodu i pragnienia.
8. Życie gracza zmniejsza się, gdy jest skutecznie atakowany przez przeciwników.
9. Gracz ma możliwość zmniejszenia wskaźnika głodu paczkami z jedzeniem.
10. Gracz ma możliwość zmniejszenia wskaźnika pragnienia butelkami z wodą.
11. Gracz ma możliwość zwiększenia wskaźnika życia apteczkami.
12. Gracz ma możliwość noszenia maksymalnie dwóch broni.
13. Gracz ma możliwość podnoszenia innych przedmiotów z ziemi, które trafiają do jego ekwipunku.
14. Gracz może włączyć i wyłączyć latarkę.
15. Aby podnieść broń, gracz musi mieć wolny slot broni oraz nie może mieć w ręku broni.
16. Ekwipunek ogranicza liczba slotów oraz maksymalna możliwa waga.
17. Maksymalny udźwig oraz liczbę slotów można zwiększyć poprzez plecak.
18. Istnieją trzy rodzaje plecaków w różnym stopniu rozszerzających ekwipunek.
19. Istnieją cztery rodzaje przedmiotów, trzy konsumpcyjne: apteczka, racje żywnościowe, butelka z wodą oraz dodatkowo paczka z amunicją.
20. Istnieje sześć rodzajów amunicji.

21. Paczka amunicji zawiera 30 sztuk naboi.
22. Apteczka leczy 20 punktów życia w ciągu 5 sekund.
23. Butelka wody odnawia 20 punktów głodu w 5 sekund.
24. Racja żywnościowa odnawia 20 punktów jedzenia głodu w 5 sekund.
25. Do strzelania potrzebny jest odpowiedni rodzaj amunicji.
26. Przeciwnicy poruszają się po mapie w sposób losowy.
27. Przeciwnik może zauważyć gracza.
28. Jeśli przeciwnik zauważy gracza, porusza się w jego kierunku.
29. Jeśli gracz znajdzie się w zasięgu ataku, przeciwnik atakuje.
30. Przeciwnik może usłyszeć gracza.
31. Jeśli przeciwnik usłyszy gracza, porusza się w kierunku wydanego dźwięku.

3.4. Wymagania niefunkcjonalne

W tym rozdziale przedstawione będą wymagania niefunkcjonalne.

1. Do implementacji aplikacji użyty zostanie silnik Unity
2. Do implementacji użyty zostanie język C#
3. Aplikacja musi działać na urządzeniu mobilnym z systemem Android
4. Aplikacja musi być kontrolowana przy pomocy kontrolera Xbox
5. System Android musi być kompatybilny z technologią wirtualnej rzeczywistości
6. Gra będzie używać rozdzielczości 1920x1080
7. Gra musi posiadać menu główne z możliwością nawigowania do nowej gry, zapisu, pomocy oraz informacje o grze
8. Gra musi posiadać okno pomocy, użytkownik będzie posiadał możliwość powrotu do menu głównego.

3.5. Opis podstawowej architektury systemu

Sztuczna inteligencja zaimplementowana w grze wykorzystuje do działania automat skończony (ang. finite state machine) -jest to abstrakcyjny, iteracyjny model zachowania systemu dynamicznego oparty na tablicy dyskretnych przejść między stanami”. Innymi słowy, model ten posiada kilka stanów, między którymi możliwe są przejścia. Dobrym opisem tego zachowania jest wzorzec projektowy Stan - „jest to czynnościowy wzorzec projektowy, który umożliwia zmianę zachowania obiektu poprzez zmianę jego stanu wewnętrznego, uzależnia sposób działania obiektu od stanu, w jakim się aktualnie znajduje[13]. W grze zaimplementowany został automat z trzema możliwymi stanami: patrol, bezczynność, podążanie.

Do implementacji ekwipunku wykorzystany został wzorzec projektowy *Iterator*. „Jest to czynnościowy, obiektowy wzorzec projektowy, którego celem jest zapewnienie sekwencyjnego dostępu do podobiektów zgrupowanych w większym obiekcie”[13]. Ekwipunek zawiera w sobie listę przedmiotów, które ma przy sobie gracz oraz udostępnia interfejs do kontrolowania tej listy, jak na przykład dodawanie czy usuwanie przedmiotów.

Następnym wzorcem projektowym jest Metoda szablonowa - czynnościowy wzorzec projektowy. Jego zadaniem jest zdefiniowanie metody będącej szkieletem algorytmu. Algorytm ten może być następnie dokładnie definiowany w klasach pochodnych. Niezmienna część algorytmu zostaje opisana w metodzie szablonowej, której klient nie może nadpisać[13]. Metoda ta jest wykorzystana do przedmiotów znajdujących się w grze. Każdy przedmiot posiada komponent *Collider* oraz metodę, która go włącza. Różne przedmioty mają różne *Collider’y* jak np. *BoxCollider* czy *CapsuleCollider*, aby go

Z komentarzem [2]:

05.03.2019

Z komentarzem [3]: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Inżynieria oprogramowania: Wzorce projektowe

Z komentarzem [4]: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Inżynieria oprogramowania: Wzorce projektowe

Z komentarzem [5]: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Inżynieria oprogramowania: Wzorce projektowe

wyłączyć trzeba odwołać się do odpowiedniego komponentu, przez co niezbędne jest nadpisanie metody klasy bazowej.

Ostatnim użytym wzorcem jest Mediator - wzorzec projektowy należący do grupy wzorców czynnościowych. Mediator zapewnia jednolity interfejs do różnych elementów danego podsystemu[13]. W grze jest to klasa, kontrolująca zarówno poruszanie się gracza oraz dostęp do jego ekwipunku.

Z komentarzem [6]: Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Inżynieria oprogramowania: Wzorce projektowe

3.6. Sposób realizacji

Gra zaimplementowana jest w środowisku Unity. Jest to multiplatformowy, dynamicznie rozwijający się silnik do tworzenia gier. Unity daje możliwość szybkiej implementacji zarówno w trybie dwuwymiarowym, jak i trójwymiarowym. Wcześniej silnik używał języków Boo, JavaScript oraz C#, przy czym z dwóch pierwszych zrezygnowano na korzyść C#. Unity zawiera w sobie dwa silniki fizyczne, dla gier dwuwymiarowych jest to Box2D, natomiast dla gier trójwymiarowych jest to silnik NVIDIA PhysX. W skład narzędzi Unity wlicza się również narzędzie do tworzenia interfejsu użytkownika, umożliwiające tworzenie zaawansowanych interfejsów szybko i intuicyjnie. Do zalet Unity można zaliczyć szybko rozwijający się *AssetStore*, gdzie można znaleźć i pobrać wiele przydatnych narzedzi jak i różnych modeli, dostarczanych zarówno przez oficjalnych twórców silnika jak i ciągle rozwijającą się społeczność. Do korzystania z wirtualnej rzeczywistości wykorzystano właśnie takie narzędzie. Alternatywą dla Unity jest *UnrealEngine*, który jest bardziej zaawansowany i lepszy graficznie, zdecydowano się na Unity. Wybór ten jest uwarunkowany dużo mniejszym progiem wejścia, przez co znacząco skrócił się czas implementacji gry oraz znacząco większym community.

Do wykonania niektórych modeli trójwymiarowych wykorzystano program Blender3D. Użycie tego oprogramowania uwarunkowane było przede wszystkim kompatybilnością modeli z silnikiem Unity. Łatwość obsługi również była ważnym powodem podczas decyzji wyboru oprogramowania. Kolejnym argumentem decydującym była darmowość i możliwość korzystania ze stworzonych modeli w projektach komercyjnych

Jako język do implementacji oprogramowania wykorzystano C#. Był to wybór oczywisty jako, że silnik Unity mocno wspiera ten język.

4. Projekt gry survival w wirtualnej rzeczywistości

4.1. Wprowadzenie

Poprzedni rozdział w całości poświęcony został założeniom projektowym. Opisane w nim zostały nie tylko wymagania funkcjonalne oraz niefunkcjonalne, ale również zarysowano ogólna architekturę systemu oraz przedstawiono sposób realizacji oprogramowania w ramach pracy. Dzięki przyjętym założeniom możliwe jest rozpoczęcie pracy nad tym rozdziałem. Jest to ważny rozdział podczas pisania pracy dyplomowej, ponieważ przed rozpoczęciem właściwej implementacji, należy najpierw przygotować oraz odpowiednio przeanalizować projekt całej gry. Pierwszym krokiem przy projektowaniu powinien być diagram przypadków użycia, który stanowić będzie podstawę, na bazie której napisane zostaną przypadki użycia. Dzięki nim zostaną uwidocznione wymagania, jakie implementowana gra musi spełnić. Następny podrozdział informować będzie o interfejsie użytkownika, który jest nieodłącznym elementem całej gry. Na koniec przedstawiona zostanie architektura gry oraz inne elementy ze strony implementacyjnej projektu.

4.2. Przypadki użycia

W niniejszym rozdziale przedstawione zostaną, historyjki użytkownika, które muszą zostać zaimplementowane w prototypie gry. Historyjki te składać się będą z:

- Nazwy przypadku
- Aktorów
- Warunków wstępnych
- Warunków końcowych
- Scenariusza głównego
- Scenariusza alternatywnego (opcjonalnie)

Aktorzy, którzy występować będą w historyjkach to:

- Gracz – główny aktor wielu historyjek, jest to obiekt, którym steruje użytkownik.
- System Ekwipunku – aktor odpowiedzialny za sterowanie ekwipunkiem.
- System Gracza – aktor odpowiedzialny za sterowanie skryptami gracza.
- Przeciwnik – fizyczny obiekt przeciwnika w prototypie, steruje nim system przeciwnika
- System przeciwnika – zbiór skryptów odpowiedzialny za sterowanie przeciwnikiem.

Historyjki użytkownika:

Tab. 6.1. Historyjka wyrzucenie przedmiotu

Nazwa przypadku	Wyrzucenie przedmiotu
Aktorzy	Gracz, System Ekwipunku
Warunki wstępne	Gracz posiada przedmioty w ekwipunku
Warunki końcowe	Przedmiot usunięty z ekwipunku
Scenariusz główny	1. Gracz otwiera okno ekwipunku 2. Gracz wybiera przedmiot z ekwipunku 3. Otwiera się okno kontekstowe 4. Gracz wybiera Wyrzuć

Tab. 6.2. Historyjka użycie przedmiotu

Nazwa przypadku	Użycie przedmiotu
Aktorzy	Gracz, System Ekwipunku
Warunki wstępne	Gracz posiada przedmioty w ekwipunku
Warunki końcowe	Przedmiot usunięty z ekwipunku, odpowiedni efekt dla przedmiotu zostaje użyty na graczu
Scenariusz główny	1. Gracz otwiera okno ekwipunku 2. Gracz wybiera przedmiot z ekwipunku 3. Otwiera się okno kontekstowe 4. Gracz wybiera Użyj.

Tab. 6.3. Historyjka podniesienie przedmiotu

Nazwa przypadku	Podniesienie przedmiotu
Aktorzy	Gracz, System Ekwipunku
Warunki wstępne	Gracz w zasięgu podniesienia przedmiotu
Warunki końcowe	Przedmiot znajduje się w ekwipunku
Scenariusz główny	1. Gracz próbuje podnieść przedmiot 2. System sprawdza czy jest miejsce w ekwipunku 3. System dodaje przedmiot do ekwipunku
Scenariusz alternatywny	3.A Limit obciążenia lub limit przedmiotów osiągnięty nie można podnieść przedmiotu. 3.A.1 Wróć do kroku 1

Tab. 6.4. Historyjka podniesienie broni

Nazwa przypadku	Podniesienie broni
Aktorzy	Gracz, System Gracza
Warunki wstępne	Pusty slot na broń
Warunki końcowe	Gracz posiada w slocie broń
Scenariusz główny	1. Gracz podchodzi do broni 2. Gracz próbuje podnieść broń 3. Gracz podnosi broń 4. System przypisuje broń do slotu
Scenariusz alternatywny	3.A Gracz nie podnosi broni 3.A.1 Wróć do kroku 2 scenariusza drugiego

Tab. 6.5. Historyjka przeładowanie broni

Nazwa przypadku	Przeładowanie broni
Aktorzy	Gracz, System Gracza
Warunki wstępne	Broń znajduje się w aktywnym slocie
Warunki końcowe	Broń posiada naboje w magazynku
Scenariusz główny	1. Gracz przeładowuje broń 2. System sprawdza czy jest dostępna amunicja 3. System aktualizuje ilość naboi w magazynku 4. System aktualizuje ilość naboi w ekwipunku
Scenariusz alternatywny	3.A Gracz może nie posiadać amunicji 3.A.1 Zakończ

Tab. 6.6. Historyjka zmiana trybu ognia

Nazwa przypadku	Zmiana trybu ognia
Aktorzy	Gracz, System Gracza
Warunki wstępne	Broń znajduje się w aktywnym slocie
Warunki koncowe	Broń zmieniła tryb ognia
Scenariusz główny	1. Gracz zmienia tryb ognia 2. System zmienia tryb ognia

Tab. 6.7. Historyjka trafienie przeciwnika

Nazwa przypadku	Trafienie przeciwnika
Aktorzy	Gracz, Przeciwnik, System Przeciwnika
Warunki wstępne	Broń znajduje się w aktywnym slocie, magazynek nie jest pusty
Warunki koncowe	Zmniejszone aktualne życie przeciwnika
Scenariusz główny	1. Gracz strzela w kierunku Przeciwnika 2. System Przeciwnika wykrywa kolizje z pociskiem. 3. System przeciwnika odejmuje aktualne życie
Scenariusz alternatywny	2.A System przeciwnika nie wykrywa kolizji. 2.A.1 Zakończ

Tab. 6.8. Historyjka zmiana stanu przeciwnika na Patrol 1

Nazwa przypadku	Zmiana stanu Przeciwnika na Patrol ze stanu Idle
Aktorzy	Przeciwnik, System Przeciwnika
Warunki wstępne	Przeciwnik znajduje się w stanie Idle, skończył się czas stanu Idle.
Warunki końcowe	Przeciwnik znajduje się w stanie Patrol
Scenariusz główny	1. System zmienia stan na Patrol 2. System informuje Przeciwnika o stanie 3. Przeciwnik wykonuje czynności ze stanu Patrol

Tab. 6.9. Historyjka zmiana stanu na Patrol 2

Nazwa przypadku	Zmiana stanu Przeciwnika ze stanu Chase na Patrol
Aktorzy	Przeciwnik, System Przeciwnika
Warunki wstępne	Przeciwnik znajduje się w stanie Chase, Gracz znajduje się poza polem widzenia Przeciwnika
Warunki końcowe	Przeciwnik znajduje się w stanie Chase
Scenariusz główny	1. System zmienia stan na Patrol 2. System informuje Przeciwnika o stanie 3. Przeciwnik wykonuje stan Patrol

Tab. 6.10. Historyjka zmiana stanu na Idle

Nazwa przypadku	Zmiana stanu Przeciwnika na Idle ze stanu Patrol
Aktorzy	Przeciwnik, System Przeciwnika
Warunki wstępne	Przeciwnik nie znajduje się w stanie Patrol, Przeciwnik dotarł do pozycji docelowej
Warunki koncowe	Przeciwnik znajduje się w stanie Idle
Scenariusz główny	1. System zmienia stan na Idle 2. System informuje Przeciwnika o stanie 3. Przeciwnik wykonuje czynności ze stanu Idle

Tab. 6.11. Historyjka zmiana stanu przeciwnika na Chase

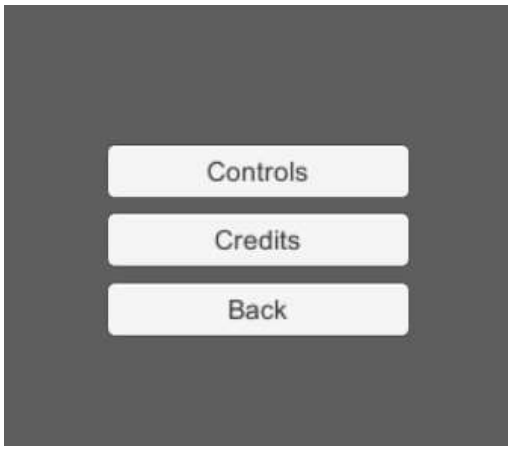
Nazwa przypadku	Zmiana stanu Przeciwnika na Chase
Aktorzy	Przeciwnik, System Przeciwnika
Warunki wstępne	Przeciwnik nie znajduje się w stanie Patrol, Przeciwnik dotarł do pozycji docelowej
Warunki koncowe	Przeciwnik nie znajduje się w stanie Chase, Gracz znajduje się w polu widzenia Przeciwnika
Scenariusz główny	System zmienia stan na Chase 2. System informuje Przeciwnika o stanie 3. System ustawia gracza jako cel 4. Przeciwnik wykonuje stan Chase

4.3. Intefrejs

W niniejszym rozdziale przedstawiony zostanie interfejs użytkownika w postaci zrzutów ekranów. Ekran który wita gracza jest zaprezentowany na Zrzucie 5.1, składa się z trzech przycisków.

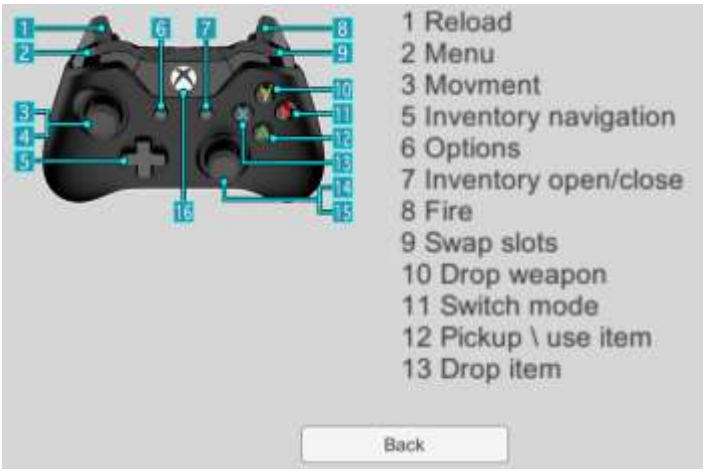
Zrzut 4.1 Menu główne gry

Po wybraniu przycisku „New game” rozpocznie się gra. W przypadku „Exit” gra zostanie wyłączona. Natomiast po kliknięciu na „Help” otworzy się nowe okno, które przedstawione jest na Zrzucie 5.2.

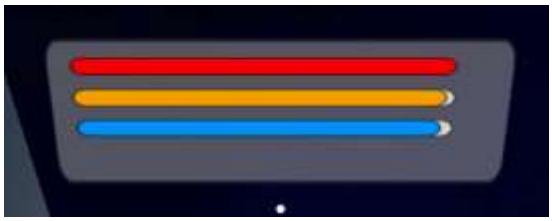


Zrzut 4.2 Menu help w grze

Przycisk „Back” wraca do głównego menu. Z kolei przyciski „Control” oraz „Credits” prowadzą do następnych okien. W pierwszym z nich przedstawiony jest system sterowania, można go zobaczyć na Zrzucie 5.3, natomiast Credits zaprezentowane na przedstawia kto zaimplementował grę.

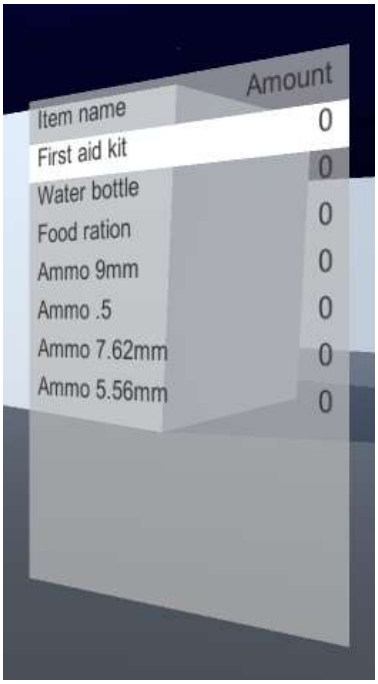


Zrzut 4.3 Schemat sterowania padem



Zrzut 4.5 Paski stanu gracza

To był cały interfejs menu głównego, teraz przedstawiony zostanie interfejs wewnątrz gry. Składa się on z unoszącym się nad głową gracza panelem z trzema paskami statusu. Widać go na Zrzut 5.5.



Zrzut 4.4 Ekwipunek gracza

Menu ekwipunku nie jest cały czas widoczne, gdyż przeszkadzałoby w grze, można je wyświetlić wciskając odpowiedni klawisz. Jest ono bardzo proste, ponieważ składa się z samego tekstu, sprawia że jest przejrzyste. Wyświetla nazwy oraz ilość posiadanych przedmiotów. Zaprezentowane zostało na Zrzucie 5.4.

Ostatnim menu jakie zostanie opisane jest widoczne na zakończenie gry. Można je zobaczyć na Zrzucie 5.7.

Zrzut 4.6 Zakonczenie gry

4.4. Ogólna architektura projektu

4.4.1. Opis struktury oprogramowania

Jedną z najważniejszych klas całego projektu jest InputManager. Jej główną odpowiedzialnością jest translacja poleceń wydawanych przy pomocy kontrolera na polecenia wydawane przez postać gracza. Realizuje to za pomocą dwóch klas, PlayerMovment oraz PlayerManager, obie te klasy posiadają instancje PlayerStats, która przechowuje aktualne informacje o graczu. Każda z nich odpowiada za różne aspekty użytkownika, na przykład klasa PlayerMovment odpowiada za poruszanie się gracza we wszystkich kierunkach oraz steruje trybami poruszania się, jak skradanie się czy skok Z kolei PlayerManager ma znacznie więcej odpowiedzialności. Steruje ona nie tylko systemem ekwipunku gracza, informuje również kiedy gracz został trafiony, poprzez odjęcie jego życia, ale również jest odpowiedzialna za kontrole aktywnej broni. PlayerManager zawiera w sobie instancje dwóch klas: PlayerWeaponSlots, która zarządza slotami na bronie gracza, oraz PlayerInventory, która odpowiada za wszystkie przedmioty poza broniami. Składa się

ona z listy instancji klasy Item oraz zajmuje się dodawaniem oraz usuwaniem podniesionych przez gracza przedmiotów.

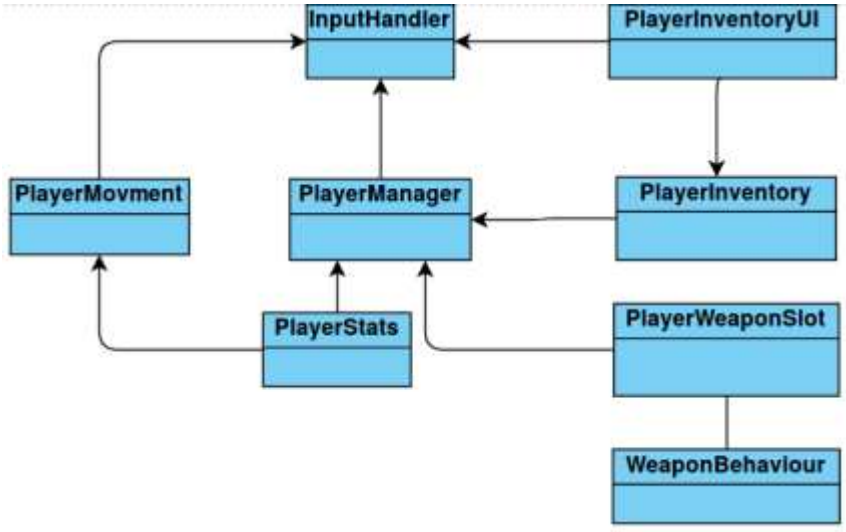


Diagram 4.1 Diagram klas przedstawiający zależności miedzy graczem a innymi klasami

Drugą najważniejszą klasą jest MobController. Jest to klasa, która odpowiada za kontrolowanie przeciwnika. Jest to prosta skończona maszyna stanów z trzema możliwymi stanami. Do kontroli przeciwnika wykorzystywane są klasy: MobPatrol, MobIdle, MobChase. Dziedziczą one po klasie MobBehavior która posiada instancje klasy MobMovment, odpowiadającej za przemieszczanie przeciwnika po labiryncie. Klasa FieldOfView wysyła informacje kiedy gracz znajdzie się w zasięgu widzenia przeciwnika.

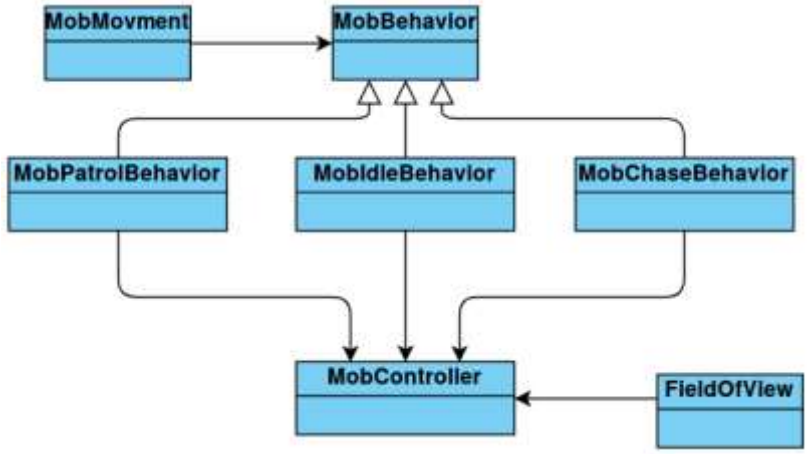


Diagram 4.2 Diagram klas przedstawia system sterowania przeciwnikiem

4.4.2. Wykorzystane wzorce

Pierwszym wzorcem projektowym użytym do realizacji projektu będzie Mediator[13]. Jest on wykorzystany do implementacji klasy PlayerManager, która będzie odpowiedzialna za sterowanie broniami oraz ekwipunkiem gracza. Wzorzec ten został tutaj zastosowany, aby wyeliminować połączenia między klasami PlayerWeaponSlot, PlayerInventory oraz IntputManager.

Do implementacji przedmiotów wykorzystano wzorzec projektowy Metoda Szablonowa[13], która wykorzystywana jest do wywołania prywatnej funkcji, która dezaktywuje dany przedmiot poprzez wyłączenie jego kolidera. Jeden przedmiot różni się koliderem od pozostałych, więc musiała zostać stworzona osobna klasa oraz metoda w niej zawarta wyłączająca odpowiedni kolider, aby dezaktywować ten przedmiot.

Następnym wzorcem zastosowanym dla tego projektu jest Iterator[13]. Klasa PlayerInventory będzie go wykorzystywać. Jest to dobry wzorzec do wykorzystania w systemie ekwipunku, gdyż daje możliwość zdefiniowania jednolitego interfejsu dla kontenera oraz sekwencyjny dostęp do listy przedmiotów.

4.4.3. Opis architektury:

Implementacja prototypu wykorzysta model architektury Model-View-Controller. Jest to „wzorzec architektoniczny służący do organizowania struktury aplikacji posiadających graficzne interfejsy użytkownika”. Największa częścią tego modelu jest element Model. Zawiera on informacje o klasach odpowiadających za przetrzymywanie danych o obiektach występujących w grze, takie jak statystyki gracza czy broni. Następny jest element Kontroler, który odpowiada za sterowanie wydarzeniami w grze. Służą do tego trzy klasy: InputManager, PlayerMovment oraz PlayerManager. Ostatnim elementem jest Widok, zapewnia on wyświetlanie oraz aktualizacje interfejsu użytkownika.

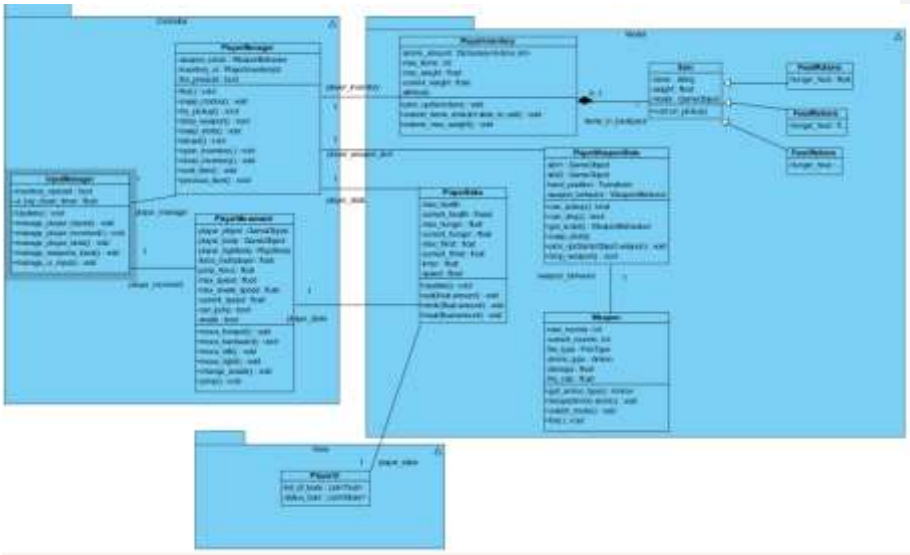


Diagram 4.3 Diagram MVC dla gry

4.5. Podsumowanie

Celem w niniejszym rozdziale była analiza założeń z rozdziału 4. oraz stworzenie projektu aplikacji do implementacji. Korzystając z wyników prac projektowych zaprezentowanych w rozdziale, możliwe jest rozpoczęcie pracy nad implementacją prototypu.

5. Implementacja

5.1. Wprowadzenie

Celem niniejszego rozdziału jest przedstawienie środowiska i narzędzi programistycznych, które zostały użyte w celu implementacji prototypu. Rozdział ten zawiera również opis architektury wysokiego poziomu i przedstawiony sposób instalacji oprogramowania.

5.2. Wykorzystane środowiska i narzędzia programistyczne

W tym podrozdziale opisane są narzędzia programistyczne oraz środowiska, które zostały wykorzystane w celu zaimplementowania prototypu gry.

Pierwszym i najbardziej wykorzystywanym narzędziem jest silnik Unity, choć można go również uznać za środowisko, ponieważ używa wiele zintegrowanych bibliotek. Do implementacji wykorzystano wiele z jego komponentów takich jak.

- Fizyka ciał
- Kolizje
- Odtwarzanie dźwięków

Ponadto dzięki użyciu assetu GoogleVR & GearVR[14], czyli assetów udostępniających komponenty i biblioteki do obsługi wirtualnej rzeczywistości, możliwe było szybkie i łatwe wykorzystanie technologii wirtualnej rzeczywistości w projekcie.

Następnym narzędziem, które wykorzystano do implementacji jest VisualStudioCode, jest to darmowy edytor kodu, został głównie wybrany ze względu na łatwy i intuicyjny w obsłudze interfejs oraz niewielki rozmiar, w przeciwieństwie do VisualStudio, które zajmuję znacznie więcej miejsca na dysku.

5.3. Opis architektury wysokiego poziomu

W tym rozdziale przedstawiona jest architektura wysokiego poziomu, czyli diagram pakietów, komponentów oraz rozmieszczenia.

System podzielony jest na sześć części jak widać na diagramie 6.1

- Input
- Items
- Player
- Weapon
- Inventory
- Mob

Pierwszy pakiet, który opisze to „Pakiet Input” jest odpowiedzialny za odbieranie poleceń wysyłanych przez użytkownika systemu, następnie przekazuje je dalej do klasy gracza lub ekwipunku

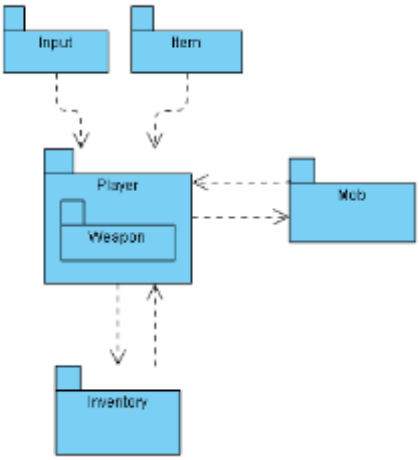


Diagram 5.1 Diagram pakietów, komunikacja gracza z innymi komponentami

Pakiet Items jest reprezentacją przedmiotów występujących w grze, są to apteczki, racje żywnościowe oraz butelki z wodą. Centralnym pakietem jest „Player” ma on najwięcej powiązań z pozostałymi pakietami, ponieważ gra skupia się w głównej części właśnie na postaci gracza. To postać odpowiedzialna jest za zbieranie przedmiotów czy otrzymywanie obrażeń. Ważną funkcjonalnością tego pakietu jest również wyposażenie gracza w broń oraz czynności jakie można z nią wykonać takie jak: przeładowanie czy zmiana trybu strzału. Następną opisanym pakietem jest „Weapon”. Odpowiada on głównie za statystyki poszczególnych broni, typ używanej amunicji czy jak jest ona obsługiwana. Pakiet Inventory służy do zarządzania ekwipunkiem gracza, trafiają do niego przedmioty, które gracz podniesie podczas rozgrywki. Ostatnim pakietem jest „pakiet Mob”, odpowiada za sterowanie przeciwnikiem podczas rozgrywki, wykorzystuje do działania automat skończony z trzema stanami.

W tym akapicie przedstawione są dwa diagramy, pierwszym jest zarządzanie graczem oraz jego ekwipunkiem, a następnie diagram sterowania przeciwnikiem,Diagram 6.2 składa się z trzech głównych komponentów:

- Player
- InputManager
- Inventory

Player jest komponentem, który odpowiada za sterowanie graczem, czyli poruszanie się, zbieranie przedmiotów oraz obsługiwanie trzymanej w ręku broni. Input Manager jest komponentem odpowiedzialnym za odbieranie sygnałów od użytkownika. Komponent ten łączy się z komponentem Player, poprzez udostępniony interfejs tego drugiego. Interfejs zawiera zbiór funkcji, które odpowiadają za reakcje gracza takie jak, podniesienie przedmiotu, ruch do przodu czy strzał. InputManager odpowiada również za zarządzanie ekwipunkiem poprzez udostępnione mu funkcje komponentu Inventory. Odpowiada on za

wyświetlanie aktualnego stanu ekwipunku oraz zarządzanie nim. Ostatnią omówioną zależnością jest ta łącząca Player oraz Inventory, polega ona głównie na dodawaniu

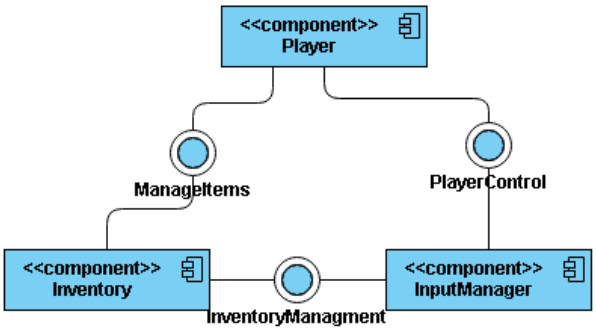


Diagram 5.2 Diagram komponentów, zarzadzanie ekwipunkiem

podnoszonego przez gracza przedmiotu do spisu przedmiotów w ekwipunku.

Na Diagramie 3 znajduje się diagram przedstawiający sterowanie przeciwnikiem. Najważniejszym komponentem tutaj jest MobController, zawiera się w nim automat skończony, który w zależności od warunków otoczenia oblicza stan przeciwnika w jakim ten powinien się znaleźć. Są trzy stany

- Idle
- Patrol
- Chase

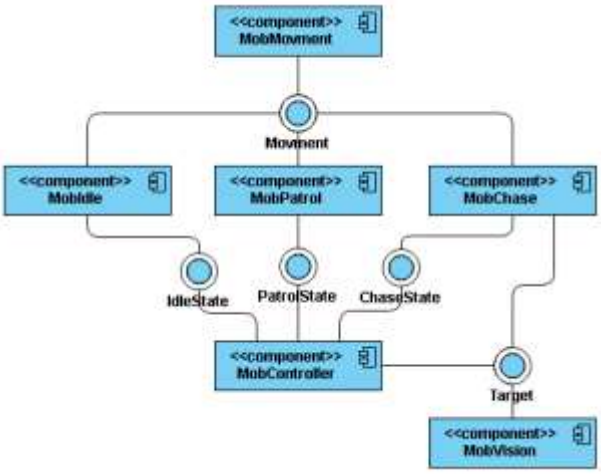


Diagram 5.3 Diagram komponentów przeciwnika

W przypadku stanu Idle przeciwnik będzie się obracał w kierunku losowo wybranego punktu, następnie po upływie do kilku sekund, czas jest losowo dobierany, przeciwnik

przechodzi w stan Patrol. W tym stanie losowany jest punkt na mapie, w zależnosci od typu przeciwnika istnieją różne sposoby wybierania punktu do którego ma się udać przeciwnik, następnie przeciwnik porusza sie w jego kierunku az do osiągnięcia celu. Po dotarciu do celu stan zmieniany jest na Idle. Ostatnim stanem jest Chase, aktywuje się gdy w polu widzenia przeciwnika pojawi się gracz, w tym stanie przeciwnik będzie podążał za graczem, a gdy go dogoni zaatakuje. Po straceniu gracza z pola widzenia przeciwnik przejdzie w stan Patrol. Za obsługę stanów odpowiadają komponenty:

- MobPatrol
- MobIdle
- MobChase

Komponenty te korzystają z funkcji udostępnianych przez komponent MobMovment, który odpowiada za poruszanie przeciwnikiem. Ostatnim komponentem jest MobView, odpowiada on za pole widzenia przeciwnika oraz informuje, kiedy przeciwnik znajdzie sie w tym polu. Udostępnia on tą informacje komponentowi MobController.

5.4. Instalacja oprogramowania

Zainstalować aplikację można tylko i wyłącznie na urządzeniach posiadąjacych system operacyjny Android. Do instalacji oprogramowania wymagany jest plik apk, zawierający prototyp. Rozpoczęcie instalacji następuje poprzez wybranie pliku apk. Po rozpoczęciu instalacji należy postępować zgodnie z krokami, które się wyświetlają podczas jej trwania

5.5. Testy

Testy przeprowadzone zostały manualnie przez autora, podczas implementacji prototypu. Jest to najprostszy sposób testowania oprogramowania, jednak jest on niewystarczający, aby zlokalizować wiele usterek, które mogą nie zostać znalezione. Dobrym sposobem testowania są testy jednostkowe, jednak w przypadku złożonej gry jest trudne sprawdzić każdą możliwą funkcje, ponadto w przypadku testów jednostkowych czas wytwarzania oprogramowania znacząco się wydłuża.

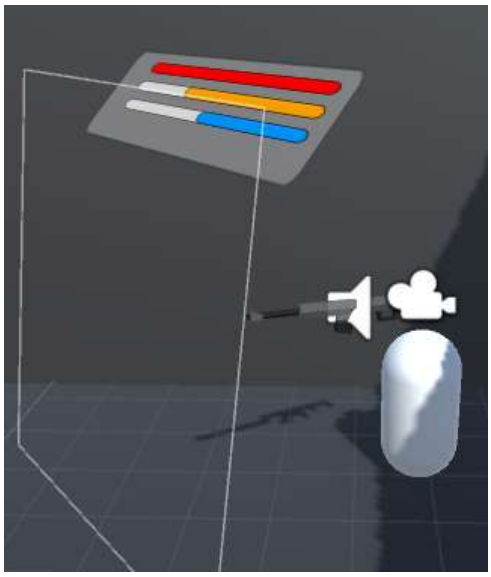
5.6. Podsumowanie

Celem w niniejszym rozdziale było przybliżenie tego, w jaki sposób autor zaimplementował prototyp, jak powinien zostać zainstalowany oraz jakie testy zostały przeprowadzone w celu sprawdzenia działania prototypu.

6. Podsumowanie pracy

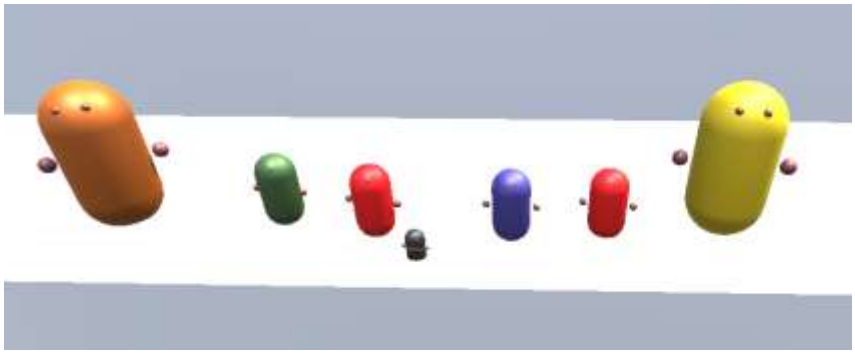
6.1. Wykonane prace

W ramach pracy wykonano grę survival w technologi wirtualnej rzeczywistości przy użyciu silnika Unity. Najważniejszą częścią gry jest gracz, bez niego gra nie miałaby sensu. Na Zrzucie 7.1 przedstawiony jest gracz trzymający broń.



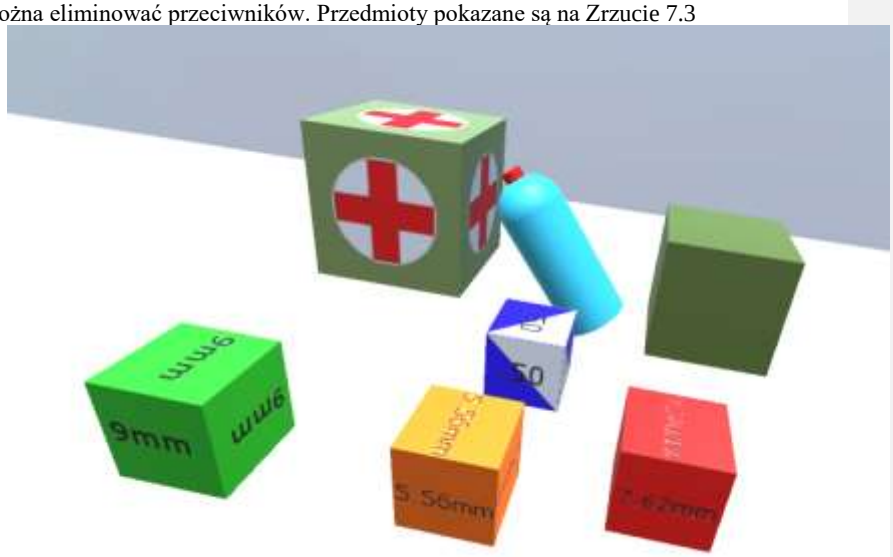
Zrzut 6.1 Obiekt gracza z bronią

Najwięcej czasu potrzeba było, aby zaimplementować przeciwników. Po napisaniu logiki przeciwników pozostało tylko skonfigurowanie ich i zrobienie drobnych zmian kosmetycznych. Na Zrzucie 7.2 przedstawieni są przeciwnicy występujący w grze.



Zrzut 6.2Przeciwnicy występujący w grze

Czym byłaby gra bez przedmiotów, gdy gracz otrzyma obrażenia, to musi się uleczyć, a gdy jest głodny zjeść. Również ważnym zasobem w grze jest amunicjia, bez której nie można eliminować przeciwników. Przedmioty pokazane są na Zrzucie 7.3



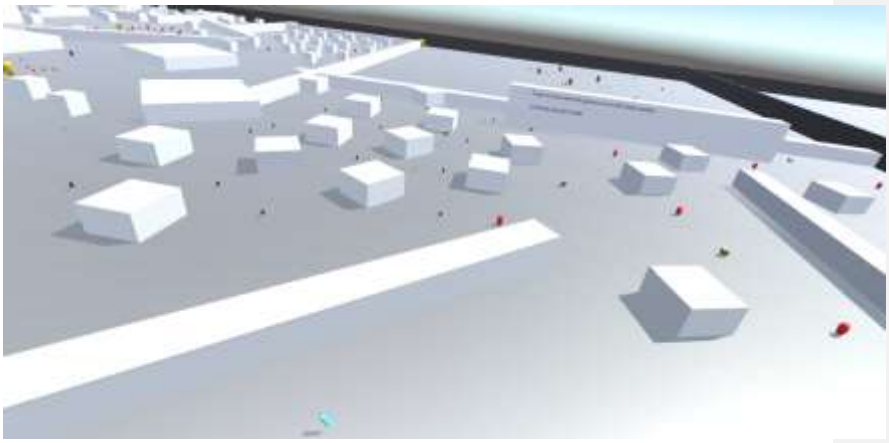
Zrzut 6.3 Przedmioty wykorzystane w grze

Każda gra survival zawiera jakąś broń, w przypadku prototypu jest to broń palna różnego rodzaju. Wszystkie bronie różnią się od siebie kilkoma parametrami takimi jak obrażenia, pojemność magazynka czy szybkością strzału. Zrzut 7.4 pokazana jest broń jaka występuje w grze

Zrzut 6.4 Modele broni z gry

Na koniec wymagany był poziom, na którym znajdą się pozostałe zaimplementowane elementy gry. Zrzuty 12 oraz 13 przedstawiają wygląd poziomu z lotu ptaka.

Zrzut 6.5 Widok z lotu ptaka perspektywa pierwsza



Zrzut 6.6 Widok z lotu ptaka perspektywa druga

6.2. Realizacja celu pracy

Dzięki zaimplementowaniu gry survival w technologi wirtualnej rzeczywistości przy użyciu silnika Unity, cel pracy został zrealizowany.

6.3. Wnioski

Ważną decyzją było rozpoczęcie prac zgodnie z procesem wytwarzania oprogramowania, dzięki temu cały proces implementacji prototypu był dobrze przemyślany oraz zaplanowany. Jednak jak w większości projektów części rzeczy nie da się zaplanować i produkt końcowy znacząco różni się od początkowych założeń, w tym wypadku przydatna jest metodyka wytwarzania oprogramowania AGILE. W projekcie brakuje testów jednostkowych, nie są one niezbędne, a jednak stają sie przydatne, gdy projekt się powiększa

6.4. Sugestie odnośnie dalszych prac

W niniejszym podrozdziale przedstawione zostaną pomysły autora na dalszy rozwój pracy. Pierwszym pomysłem do kontynuacji pracy jest tryb sieciowy pozwalający na gry kooperacyjne lub starcia graczy między sobą. Dzięki temu znacząco mogłoby wzrosnąć zainteresowanie grą i poszerzyć nieco szeregi graczy wirtualnej rzeczywistości. Dodanie trybu sieciowego pociąga za sobą zapotrzebowanie na dodatkowe poziomy, ten problem można rozwiązać poprzez zaprojektowanie kilku trudnych poziomów lub dodać proceduralnie tworzone labirynty, dzięki temu pojawi się różnorodność w grze.

Bibliografia

[1] "Poczatki wirtualnej rzeczywistości", 2019 [Online] Available: https://www.miastogier.pl/wiki,strona-2592,historia_vr.html

[2] "Porównanie googli VR", 2019 [Online] Available: https://tabliczni.pl/gadzety/porownanie-gogli-vr-pojedynek/.

[3] "Silnik Unity", 2019 [Online] Available: https://unity.com

[4] "Gra Unturned", 2019 [Online] Available: https://www.gry-online.pl/gry/unturned/z13dc7

[5] "Gra Superhot VR", 2019 [Online] Available: https://www.gry-online.pl/gry/superhot-vr/zf4b63

[6] "Gra Player's Unknown Battlegrounds", 2019 [Online] Available: www.pubg.com/

[7] "Gra ApexLegends", 2019 [Online] Available: https://www.gry-online.pl/gry/apex-legends/z25661

[8] "Gra Uncrowded", 2019 [Online] Availabie: https://mmorpg.org.pl/gra/uncrowded

[9] "System operacyjny Android", 2019 [Online] Availabie: https://en.wikipedia.org/wiki/Android_(operating_system)

[10] "Język C#", 2019 [Online] Availabie: https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

[11] "Informacje o Bluetooth", 2019 [Online] Availabie: https://www.bluetooth.com/bluetooth-technology/radio-versions//

[12] "Informacje o silniku Unity", 2019 [Online] Availabie: https://unity3d.com/unity

[13] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Inżynieria oprogramowania: Wzorce projektowe, Helion, 2010

[14] "GoogleVR & GearVR", 2019 [Online] Availabie: https://assetstore.unity.com/packages/tools/utilities/googlevr-gearvr-camera-setup-67019

[15] Jeremy Gibson Bond, Projektowanie gier przy użyciu srodowiska Unity i języka C#. Od pomysłu do gotowej gry. Wydanie II, Helion, 2018

[16] Ewa i Jacek Ross, Unity i C#. Podstawy programowania gier, Helion, 2018

[17] Jeremy Bailenson, Wirtualna rzeczywistość. Doznanie na żądanie, Helion, 2019

[18] Alan Thorn, Unity i Blendr. Praktyczne tworzenei gier, Helion, 2015

Spis rysunków

Spis table

Załącznik