

**UNIWERSYTET JAGIELŁOŃSKI
Wydział Matematyki i Informatyki**

kierunek: Informatyka
specjalność: Informatyka Stosowana

Klaudia Olejniczak

Teiru MMORTS (Massively multiplayer online real-time strategy)
Projekt i implementacja interfejsu graficznego (GUI).

Kraków 2014

Abstrakt

Graficzny interfejs użytkowy to kluczowy element dla każdego programu, pośredniczący pomiędzy programem a użytkownikiem. Szczególnie ważny w przypadku programów biznesowych przeznaczonych dla osób które niekoniecznie posiadają doświadczenie informatyczne.

Niniejsza praca przedstawia etap projektowania i implementacji użytkowego interfejsu graficznego przeznaczonego do wieloosobowej gry komputerowej czasu rzeczywistego.

Na początku mojej pracy postaram się przybliżyć państwu pokrótkę obraz całego projektu, następnie zaprezentuje trochę informacji teoretycznych, skupię się głównie na języku UML i jego użytecznością w procesie modelowania systemu. Treść następnego rozdziału ma za zadania zapoznać państwa z modelem UML systemu pod kątem interfejsu graficznego, którego proces implementacji zostanie później również przedstawiony. Na samym końcu mojej pracy zostaną przedstawione efekty powstałe w wyniku zaprojektowania i zaimplementowania w postaci zrzutów ekranu z krótkimi opisami oraz podsumowanie .

Abstract

Graphical user interface is a key element, the intermediary between the program and the user. It is particularly important in the case of business programs designed for people who do not necessarily have computer experience.

This presentation shows the design and implementation phases of the GUI utility designed for multiplayer real-time computer games.

I would like to begin with presenting you a brief picture of the entire project, followed with theoretical information, I will be focusing mainly on the UML and its usefulness in the process of modeling the system. The content of the next chapter is meant to familiarize you with the UML state model system for graphical interface, whose implementation process will also be presented. I will finish with the effects resulting from design and implementation as screenshots with short descriptions and a summary.

Spis treści

1. Wstęp.	3
2. Krótka charakterystyka gry – Teiru MMORTS.	4
3. Teoria	5
3. Model interfejsu graficznego.	8
3.1 Diagram przypadków użycia.	9
3.2 Diagram klas.	13
3.3 Diagram sekwencji.	15
3.4 Diagram stanów.	
3.5 Diagram komunikacji/kooperacji	
4. Implementacja.	
4.1 Unity	
4.2 ...	
5. Screeeny i opis.	
5.1 Ekran logowania.	
5.2 Ekran rejestracji.	
5.3 Ekran menu głównego i opcje.	
5.4 Ekran wyboru i tworzenia postaci.	
6. Podsumowanie.	

Bibliografia

Spis rysunków

Spis tabel

Spis załączników

Rozdział 1

Wstęp

Gry komputerowe towarzyszą nam wiernie od roku 1947, w którym to powstał pierwszy analogowy symulator pocisku rakietowego. Pierwszy pełny użytkowy interfejs graficzny został opracowany przez naukowców w Xerox PARC i używany jako główny interfejs przez komputer Xerox Alto od roku 1973. Prototyp opierał się głównie o hipertekst w formie tekstowej, jednak pierwsze pełne GUI obsługiwało oczywiście grafikę.

Wraz z rozwojem komputerów, jak i pojawiением się internetu, obydwie te rzeczy zdecydowanie ewaluowały, dając nam coraz to większe możliwości, jak i również stawiając coraz to większe wymagania przed deweloperami. Aktualnie żyjemy w dobie internetu, gdzie rosnącą popularność odnotowują gry wieloosobowe (ang. multiplayer), pozwalające na rywalizację wielu graczy z różnych stron świata.

Istnieje wiele sposobów tworzenia interfejsów użytkowych dla aplikacji. Projekt opisany w pracy wykorzystuje środowisko UNITY, które za pomocą potężnego silnika renderowania zintegrowanego z intuicyjnym interfejsem pozwala kreować gry typu 3D i 2D na wiele platform. Obecnie gry stworzone w oparciu o UNITY dominują rynek.

Interfejs graficzny jest warstwą, która pośredniczy pomiędzy użytkownikiem a programem. Jest jedyną rzeczą z której użytkownik ma jakąkolwiek interakcje. Stworzenie odpowiedniego GUI dla aplikacji nie jest prostym zadaniem, dlatego w tej kwestii bardzo ważne jest określenie grupy odbiorców do którego dana aplikacja będzie skierowana. Jak wiadomo wygląd i intuicyjność naszej aplikacji mocno decyduje zarówno o komforcie w użytkowaniu jak i jej popularności. Często możemy spotkać aplikacje które nie były zbyt dobrze przemyślone, a ich użytkowanie jest zdecydowanie problematyczne i nieefektywne, zrażając użytkownika. W takich przypadkach wiele naprawdę wartych uwagi aplikacji traci wsparcie.

Na początku mojej pracy postaram się przybliżyć państwu pokrótce obraz całego projektu, wymagania i funkcjonalność jaką ustaliliśmy wspólnie z zespołem. Następnie zaprezentuję trochę teorii, skupię się głównie na języku UML i jego użytecznością w procesie modelowania systemu.

Treść następnego rozdziału ma za zadania zapoznać państwa z modelem UML systemu pod kątem interfejsu graficznego, którego proces implementacji zostanie później również przedstawiony. Pod koniec tego rozdziału znajdzie się krótki rozdział, związany z wnioskiem jaki nasunął mi się podczas tworzenia modeli dla warstwy interfejsu użytkownika. Na samym końcu mojej pracy zostaną przedstawione efekty powstałe w wyniku zaprojektowania i zaimplementowania w postaci zrzutów ekranu z krótkimi opisami oraz podsumowanie tego etapu.

Celem mojej pracy jest stworzenie interfejsu graficznego dla wieloosobowej gry czasu rzeczywistego – Teiru, poprzez projekt i implementacje, tak aby stworzyć jak najbardziej wygodne i intuicyjne środowisko dla gracza.

Rozdział 2

Charakterystyka projektu.

2.1 Krótka charakterystyka gry – Teiru MMORTS.

Gra strategiczna stworzona w przestrzeni dwuwymiarowej, oparta głównie na rozgrywkach pomiędzy graczami. Posiada tryb dla pojedynczego gracza (ang. single player) z jedną kampanią pozwalającą użytkownikowi zapoznać się z zasadami panującymi w grze oraz trybem dla wielu graczy z rozgrywką prowadzoną w sieci Internet (ang. MMO - Massively multiplayer online), który to tryb jest głównym elementem aplikacji.

Single player jest to krótki zarys fabularny mający wprowadzić gracza w najważniejsze rzeczy związane z częścią MMO, takie jak klasy postaci i ich różnice, system prowadzenia walk, zadania (ang. quests) oraz związane z nimi nagrody, takie jak złoto i EXP (ang. Experience points), poruszanie się po świecie oraz trochę informacji fabularnych . A na końcu sposób rozwijania postaci ,którzy przychodzi wraz podniesieniem poziomu postaci (ang. level up).

Gra posiada linie zadań (and. quests) , które mają za zadanie pomóc graczowi rozwijać jego postać , poprzez zdobywanie punktów doświadczenia (and. experience points) . Pojedynek są oparte o zasadę papierowych gier RPG, bazującą na turach i rzutach kostką.

Niezależnie od kampanii, każdy gracz rozpoczyna grę jedną ze stworzonych postaci . Od tej pory ma pełną kontrolę nad losami swojej postaci, zgodnie z możliwościami przedstawionymi mu w trybie single player

2.1 Technologie i narzędzia.

Klient został napisany w języku C# z wykorzystaniem środowiska graficznego Unity w wersji 4.6. Serwer został napisany w języku Java z wykorzystaniem bibliotek Netty, wersja 3.6.10 i MySql Connector, wersja 5.1.34.

2.2 Wymagania sprzętowe aplikacji:

Procesor: dowolny procesor Dual Core, minimum 1,2 GHz

RAM: minimum 512 MB

Grafika: karta graficzna z minimum 256 MB pamięci RAM oraz obsługująca DirectX w wersji minimum 9c

HDD: minimum 1GB miejsca dostępnego na dysku

Pozostałe: **dla trybu multiplayer wymagane stałe połączenie internetowe, klawiatura, myszka.**

Rozdział 3

Teoria.

Rosnąca popularność programowania obiektowego zmieniła metody modelowania systemów informatycznych. Modelowanie strukturalne zostało zastąpione przez modelowanie obiektowe. Standardem w tej dziedzinie stał się język UML (Unified Modelling Language) -- graficzny system wizualizacji, specyfikowania oraz dokumentowania składników systemów informatycznych.

3.1 Język UML

Opis systemu wykonany za pomocą języka UML jest jednoznaczny, co bardzo ułatwia napisanie kodu źródłowego w oparciu o modele. Narzędzia do modelowania obiektowego umożliwiają wygenerowanie szkieletu klas i obiektów, a po odpowiednim zintegrowaniu ze środowiskiem programistycznym -- pozwalają na dwukierunkową synchronizację modelu z kodem źródłowym.

3.2 Diagram przypadków użycia

Diagramy przypadków użycia wykorzystywane są w fazie zbierania wymagań do reprezentowania elementów funkcjonalnych systemu. Koncentrują się na zachowaniu systemu postrzeganym z zewnątrz. Przypadek użycia opisuje funkcję realizowaną przez system jako efekt widoczny dla aktora. Z kolei aktor jest personifikacją encji uwikłanej w interakcję z systemem – taką encją może być fizyczne otoczenie, bądź użytkownik.

Przypadek użycia jest abstrakcją, która opisuje wszelkie możliwe scenariusze związane z określonym elementem funkcjonalnym systemu. Scenariusz jest instancją przypadku użycia, opisującą konkretny ciąg akcji. Scenariusze używane są jako przykłady ilustrujące konkretne sytuacje — ich treść koncentruje się na zrozumiałości.

Zależności komunikacyjne

Aktor i przypadek użycia komunikują się, gdy między nimi wymieniana jest informacja. Relacja komunikacji reprezentowana jest w diagramie za pomocą *linii ciągłej* łączącej aktora z przypadkiem użycia.

Gdy jeden przypadek użycia jest częścią drugiego — czyli gdy przepływ zdarzeń w ramach jednego z nich jest częścią przepływu zdarzeń drugiego — mówimy, że przypadki te połączone są relacją zawierania. Na diagramie relacja ta odzwierciedlona jest w postaci *linii przerywanej* opatrzonej etykietą «*include*» i zakończonej strzałką skierowaną w stronę przypadku zawieranego.

Relacja rozszerzania to inny środek redukowania modelu przypadku użycia. Mając konkretny przypadek użycia, możemy rozszerzyć go o dodatkowe zdarzenia, a uzyskamy nowy, bardziej obszerny. Na diagramie relacja ta odzwierciedlona jest w postaci *linii przerywanej* opatrzonej etykietą «*extend*» i zakończonej strzałką skierowaną w stronę przypadku rozszerzanego.

Relacja dziedziczenia to trzeci mechanizm redukowania złożoności modelu. W wyniku dziedziczenia jeden przypadek użycia może stanowić konkretyzację innego, bardziej ogólnego, przez wzbogacenie go o dodatkowe szczegóły. Reprezentowany za pomocą linii ciągłej i strzałki skierowanej ku klasie po której dziedziczy.

3.3 Diagram klas

Przeznaczeniem diagramów klas jest opisywanie struktury systemu. Klasy są abstrakcjami określającymi wspólną strukturę i wspólne zachowanie określonego zbioru obiektów. Obiekty są instancjami (egzemplarzami) klas; w czasie funkcjonowania systemu instancje te mogą być tworzone, modyfikowane i niszczane. Dla każdego obiektu określony jest stan, na który składają się wartości jego atrybutów i jego połączenia z innymi obiektami.

Diagram klas jest opisem systemu w kategoriach obiektów, klas, atrybutów, operacji i skojarzeń między obiektami.

Zależności

Asocjacja reprezentuje czasowe powiązanie pomiędzy obiektami dwóch klas. Obiekty związane asocjacją są od siebie niezależne. Asocjacja jest też używana jako alternatywny (obok atrybutu) i równorzędny sposób zapisu cech klasy.

Agregacja reprezentuje relację typu całość-część, w której część może należeć do kilku całości, a całość nie zarządza czasem istnienia części.

Agregacja całkowita jest relacją typu całość-część, w której całość jest wyłącznym właścicielem

części, tworzy je i zarządza nimi.

Relacja dziedziczenie\ a tworzy hierarchię klas, od ogólnych do bardziej szczegółowych.

3.4 Diagram sekwencji

Zadaniem diagramów interakcji jest formalizacja dynamicznego zachowania się systemu i uwidocznienie komunikacji między obiektami. Okazuje się to użyteczne dla identyfikowania dodatkowych obiektów związanych z przypadkami użycia — każdy obiekt mający związek z konkretnym przypadkiem użycia nazywać będziemy obiektem uczestniczącym.

3.5 Diagram stanów

Diagram stanów opisuje zachowanie się pojedynczego obiektu, rozumiane jako przebywanie w danej chwili w określonym stanie oraz przechodzenie z jednego stanu do innego. Pod pojęciem „stanu” rozumiemy tu zbiór wartości związanych z obiektem, zaś jako „przejście” — zmianę bieżącego stanu na inny, pod wpływem spełnienia określonego warunku.

3.6 Diagram aktywności

Diagram aktywności odzwierciedla zachowanie systemu w kategoriach aktywności. Aktywności jako elementy modelowania reprezentują wykonywanie zbioru lub ciągu operacji — konkretna aktywność może być zapoczątkowana w konsekwencji zakończenia innej aktywności, udostępnienia jakiegoś obiektu czy też ogólnie wskutek wystąpienia jakiegoś zdarzenia zewnętrznego.

3.7 Diagram komunikacji/kooperacji

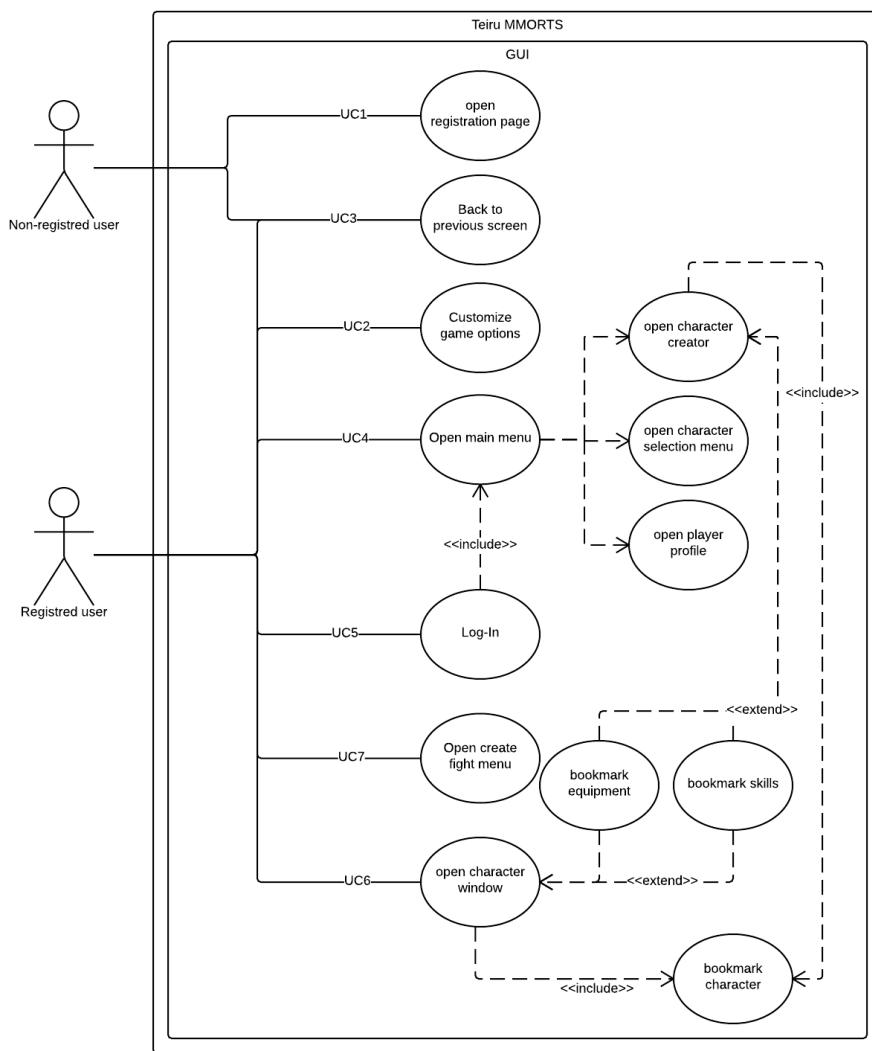
Diagram komunikacji skupia się na obiektach wchodzących w skład interakcji i wymienianymi przez nie komunikatach, natomiast w mniejszym stopniu niż diagram sekwencji (choć nadal obecnym) wskazuje na aspekt czasowy. Z tego powodu obiekty na diagramie komunikacji są umieszczone tak, aby łatwo można było opisać ich relacje pomiędzy sobą. Komunikacje są przedstawiane za pomocą linii łączących obiekty, natomiast przesyłane między obiektami komunikaty i dane są umieszczane obok tych linii. Każdy komunikat jest opatrzony etykietą liczbową, wskazującą na kolejność ich wysyłania. E

Rozdział 4

Model interfejsu graficznego.

Celem tego rozdziału jest zaprezentowanie modelu użytkowego interfejsu graficznego w grze komputerowej Teiru MMORTS.

4.1 Przypadki użycia GUI



Rys.1 Diagram przypadków użycia – GUI

Przypadek użycia	Scenariusz
<p>ID: UC1</p> <p>Nazwa: Open registration page</p> <p>Aktorzy: Non-Registered user</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. User wybiera opcje rejestracji. 2. System prezentuje formularz rejestracji w nowym oknie. 3. User wypełnia formularz. 4. System validuje dane. 5. System po udanej rejestracji informuje użytkownika o tym fakcie. <p>Wyjątki i rozszerzenia:</p> <p>3.A User nie uzupełnił wszystkich wymaganych pól w formularzu.</p> <ol style="list-style-type: none"> 1. System informuje gracza o problemie. 2. Przejdz do kroku 3. <p>4.A User próbuje zarejestrować nazwe/email, która już istnieje w bazie.</p> <ol style="list-style-type: none"> 1. System informuje gracza o problemie. 2. Przejdz do kroku 3. <p>5.A User nie przechodzi procesu uwierzytelnienia i nie zostaje zarejestrowany o czym informuje go system.</p>
<p>ID: UC2</p> <p>Nazwa: Customize game options</p> <p>Aktorzy: Registed</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. Użytkownik uaktywnia button „game options” . 2. System wyświetla okno z ustawieniami. 3. Użytkownik może dostosować opcje gry do swoich wymagań. 4. Użytkownika zapisuje swoje zmiany . 5. System zapisuje zmiany. 6. Okno opcji zostaje zamknięte automatycznie. <p>Wyjątki i rozszerzenia:</p> <p>4.A 1. User anuluje swoje decyzje poprzez przycisk „back”.</p> <ol style="list-style-type: none"> 2. Krok 6.
<p>ID: UC3</p> <p>Nazwa: Back to previous screen</p> <p>Aktorzy: Registered user, non-registered user</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. User uaktywnia przycisk „back” na dowolnym ekranie. 2. User zostanie cofnięty do poprzedniego ekranu.

<p>ID: UC4</p> <p>Nazwa: Open main menu</p> <p>Aktorzy: Registered user</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. User otwiera główne menu. 2. System wyświetla graczowi okno z menu. 3. User dokonuje akcji <p>Wyjątki i rozszerzenia:</p> <p>1.A Menu główne zostaje wyświetlone też zaraz po zalogowaniu się gracza.</p> <p>3.A Kreator postaci - Gracz zostaje odesłany do ekranu tworzenia postaci.</p> <ul style="list-style-type: none"> A 1. Gracz tworzy postać. 2. Rozpoczyna grę. B Gracz wraca do ekranu menu głównego. <p>3.B Wybór postaci – Gracz może wybrać postać którą chce grać.</p> <ul style="list-style-type: none"> A 1. Wybiera postać. 2. Rozpoczyna grę. B Gracz wraca do ekranu menu głównego. <p>3.C Profil gracza – Gracz może zobaczyć swój profil, ewentualnie edytować.</p> <ul style="list-style-type: none"> 1. Edytuje 2 A. Zapisuje efekty. B. Anuluje akcje. 3. Wraca do menu głównego.
<p>ID: UC5</p> <p>Nazwa: Log In</p> <p>Aktorzy: Registered user</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. User wybiera opcje wylogowania. 2. User loguje się do gry. <p>Wyjątki i rozszerzenia:</p> <p>1.A 1. User rezygnuje z logowania i opuszcza program. 2. Program zostaje zamknięty.</p> <p>2.A 1. User wpisuje dane do formularza 2. A. Dane są niedpoprawne, zostaje wyświetlone ostrzeżenie. B. Dane są poprawne, użytkownik zostaje przekierowany do strony głównej.</p>
<p>ID: UC6</p> <p>Nazwa: Open character window</p> <p>Aktorzy: Registered user</p>	<p>Scenariusz główny:</p> <ol style="list-style-type: none"> 1. User otwiera menu kreatora postaci. 2. System wyświetla graczowi okno z formularzem. 3. Użytkownik uzupełnia formularz i kliką „dalej”. 4. System wyświetla graczowi ekran wyboru skilli.

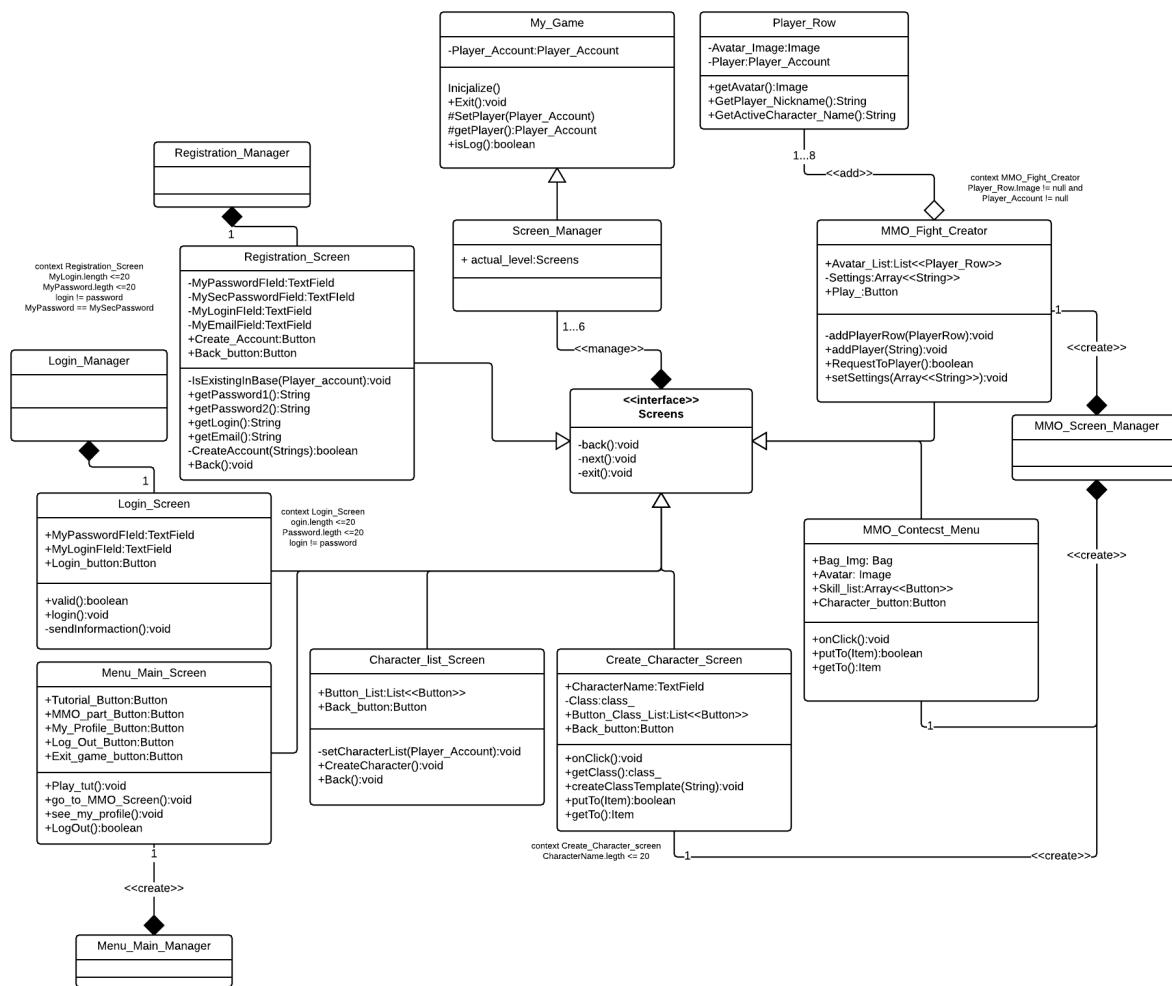
	<p>5. Gracz wybiera skille i wybiera kolejny krok 6. System wyświetla graczowi zakładkę ekipunek. 7. Gracz wybiera ekipunek. 8. Zapisuje swoją nową postać.</p> <p>Wyjątki i rozszerzenia:</p> <p>1.A 1. User rezygnuje z tworzenia postaci. 2. User zostaje przekierowany do main menu.</p> <p>2.A 1. User nie wykorzystuje wszystkich punktów statystyk. 2. System wyświetla ostrzeżenie o tym.</p> <p>8.A 1. User rezygnuje z tworzenia postaci. 2. User zostaje przekierowany do main menu.</p>
<p>ID: UC7</p> <p>Nazwa: Open create fight window</p> <p>Aktorzy: Registered user</p>	<p>Scenariusz główny:</p> <p>1. User otwiera okno kreatora walki 2. Gracz ustala drużynę. 3. W okienku formularzy wpisuje odpowiednio długość trwania tury każdego gracza. 4. Klikna przycisk „Play”</p> <p>Wyjątki i rozszerzenia:</p> <p>1.A 1. User rezygnuje z tworzenia walki. 2. User zostaje przekierowany do main menu.</p> <p>2.A 1. User dodaje nowego gracza. 2. Gracz zgadza się. 3.A Awatar i nick gracza zostaje przydzielony do odpowiedniego zespołu. 3.B Gracz przenosi zaakceptowana osobę, to innej drużyny.</p> <p>2.B 1. User dodaje nowego gracza. 2. Gracz nie zgadza się. 3. Do ekranu nie zostaje dodany nowy element.</p>

Tab. 1 Tabela do Diagramu przypadków użycia (Rys. 1).

3.3 Diagram klas dla modelu GUI

3.3.1 Diagram klas dla ogólnej budowy ekranów.

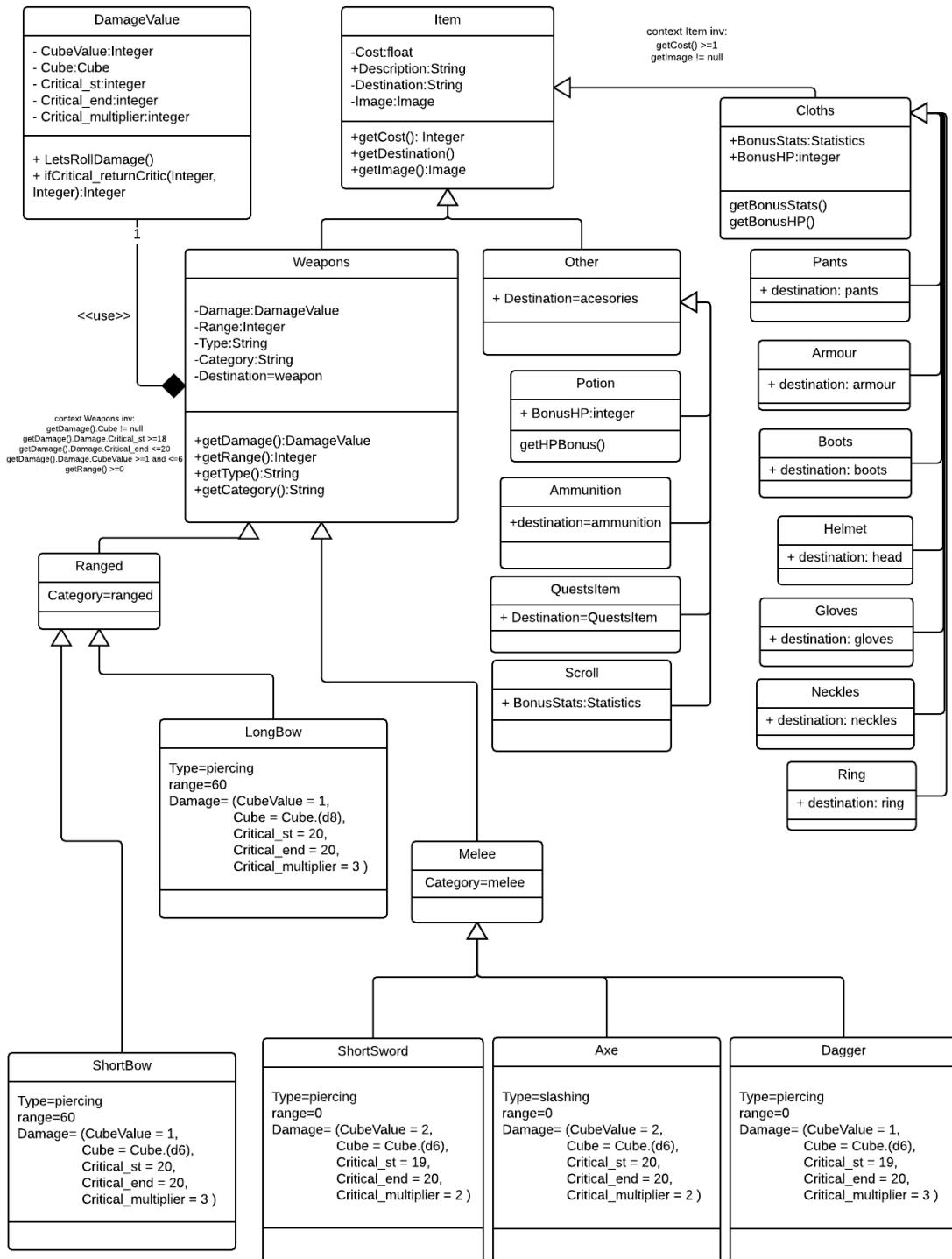
Każdy ekran programu, który ma na celu wyświetlanie elementów oraz formularzy posiada „menadżera” odpowiadającego za sterowanie zachowaniem ekranów oraz walidacje formularzy.



Rys.2 Diagram klas dla ogólnej budowy ekranów w programie.

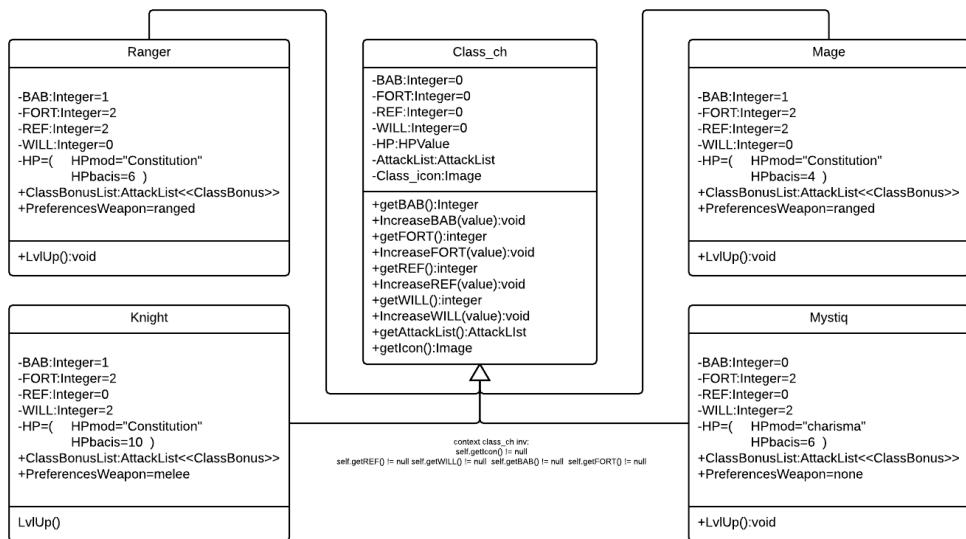
3.3.2 Diagram klas dla elementów wyświetlanego w GUI .

Elementy te są wykorzystywane w menu kontekstowym oraz kreatorze postaci.

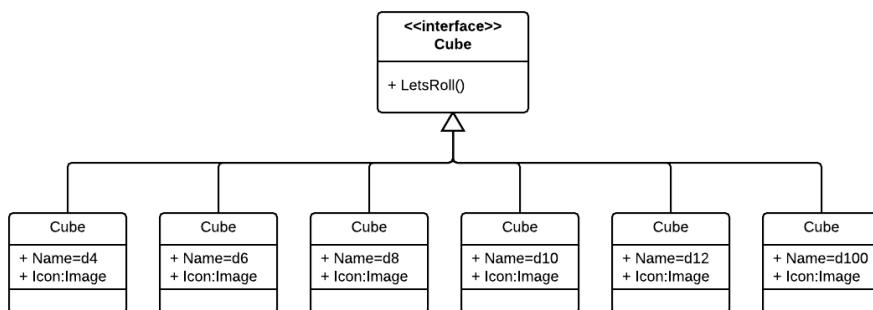


Rys.3 Diagram klas dla elementów „Item” .

Elementy te mają posłużyć wyświetlaniu elementów w GUI, na ekranach kreatora postaci, czy kreatora walk, oraz w menu kontekstowym. Mają oszczędzić żmudnego wpisywania statystyk, bądź braku możliwości wyświetlenia ich co utrudnia tworzenie postaci. W tym wypadku mowa oczywiście o klasach „Class_ch”. Za to hierarchia klasy przedmioty, bierze duży udział w tworzeniu ekwipunku postaci na ostatniej zakładce kreatora. Wyświetlane będą tam ikony przedmiotów oraz ich opisy, wraz z całą specyfikacją. Elementy kości będą się wyświetlały w menu kontekstowym w trakcie walk lub przy samych atakach, informując o obrażeniach jakie zadają, oraz jako buttony które wyzwalają akcje.



Rys.4 Diagram klas dla elementów „Class_ch” .

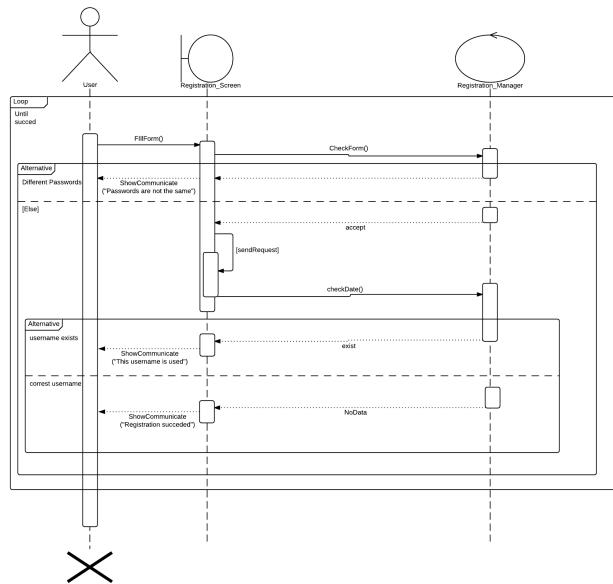


Rys.5 Diagram klas dla elementów „Dice” .

3.4 Diagram sekwencji

3.4.1 Diagram sekwencji – Rejestracja

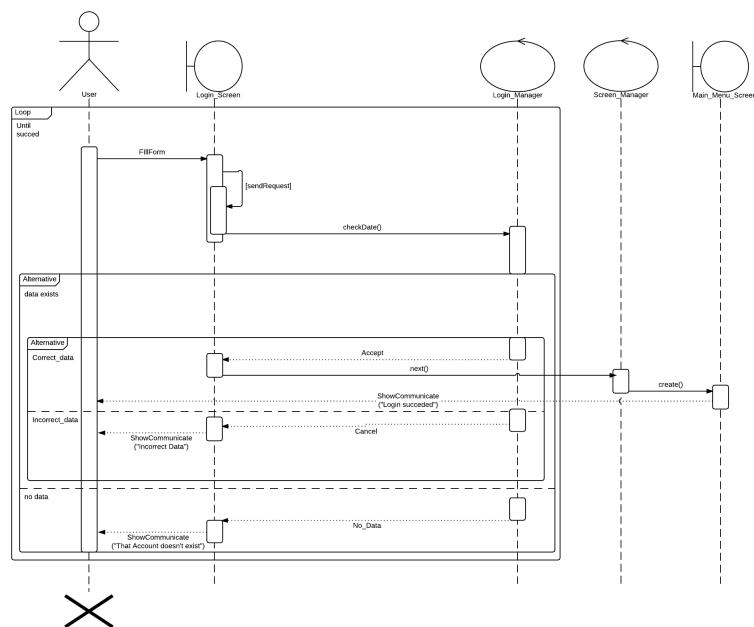
Poniżej przedstawiony jest przebieg procesu rejestracji w obrębie GUI.



Rys.6 Diagram sekwencji – Rejestracja

3.4.2 Diagram sekwencji – Logowanie

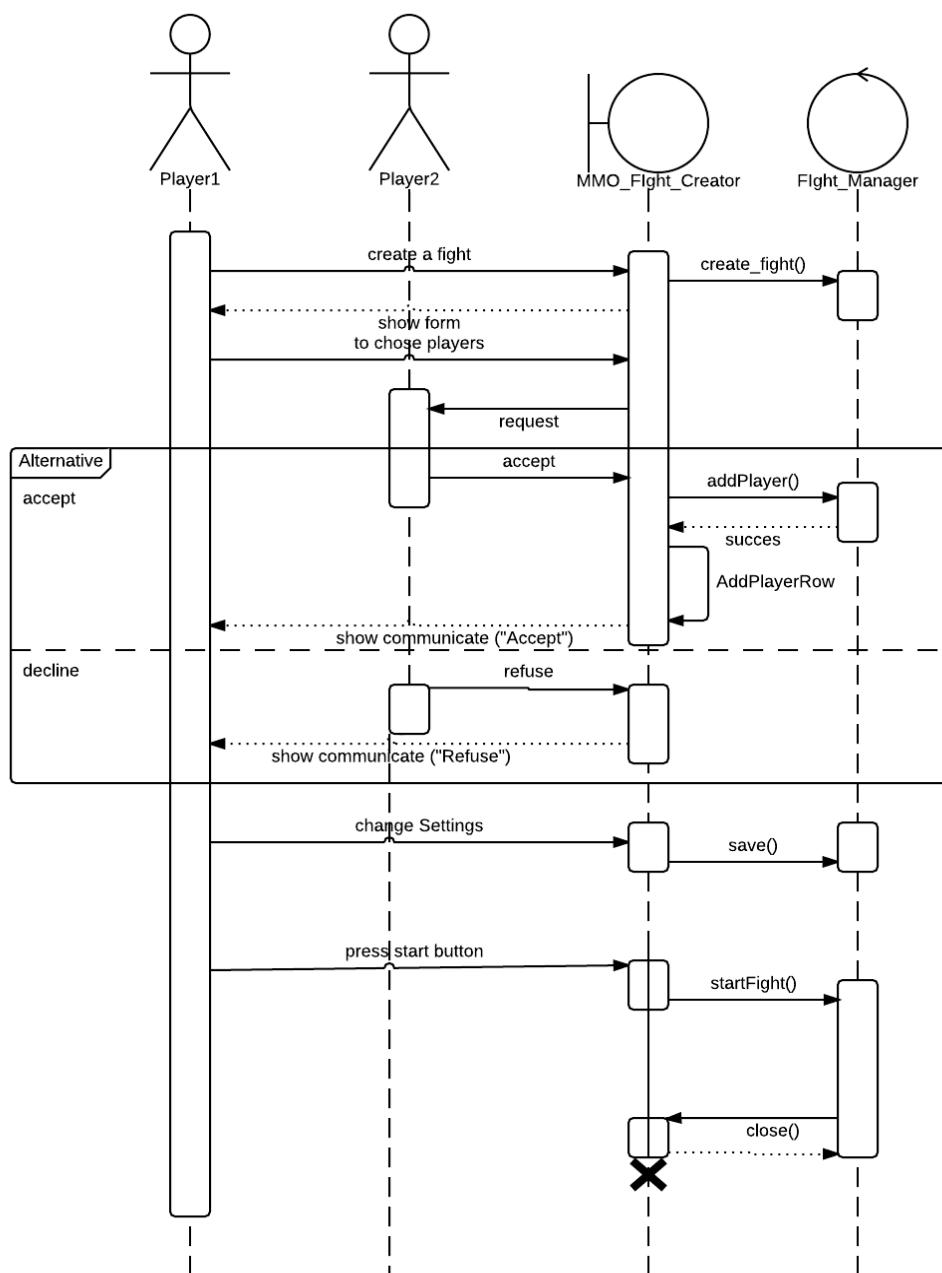
Poniżej przedstawiony jest przebieg procesu logowania w obrębie interfejsu użytkoweggo.



Rys.7 Diagram sekwencji – Login

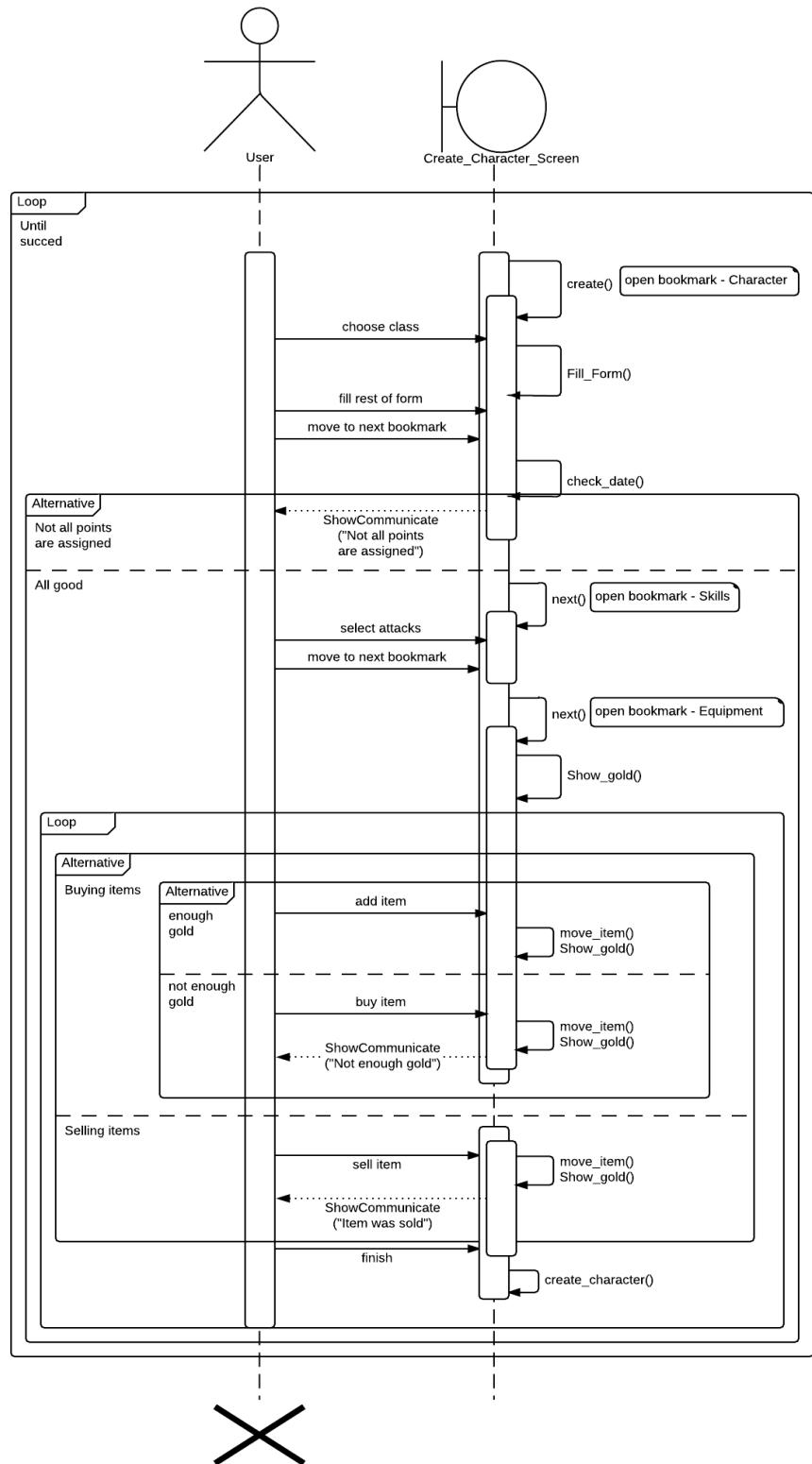
3.4.3 Diagram sekwencji – Tworzenie walki w okienku kreatora walk.

Poniżej przedstawiony jest przebieg procesu tworzenia pojedynku w obrębie interfejsu użytkowego w okienku kreatora walk.



Rys.8 Diagram sekwencji – Tworzenie walki PvP/TvT/GvG

3.4.4 Diagram sekwencji – Kreator postaci.



Rys.9 Diagram sekwencji – Kreator postaci

3.5 Diagram stanów.

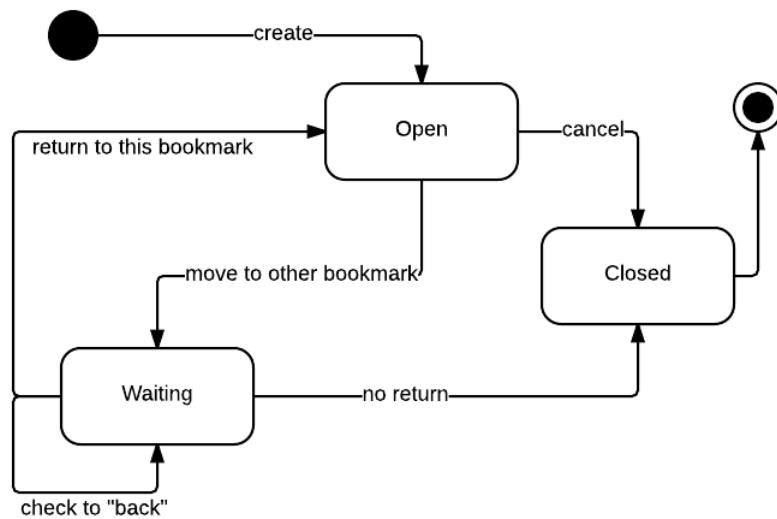
3.5.1 Diagram stanów - Okna



Rys.10 Diagram stanów – Okna

3.5.2 Diagram stanów - Zakładki w oknie kreatora postaci.

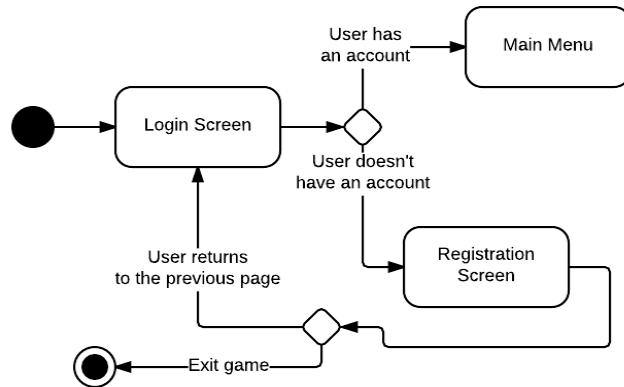
Zakładki mogą być w trzech stanach, w zależności od tego co się właśnie dzieje w oknie. Weźmy pod uwagę pierwszą zakładkę kreatora „Character”, gdy użytkownik ją używa jest ona w stanie „open”, gdy idzie do następnej przechodzi w stan „waiting”, ponieważ za pomocą przycisku „back”, ciągle można do niej wrócić. I na końcu w stanie close, gdy okno zostało zamknięte.



Rys.11 Diagram stanów – zakładki w oknie kreatora postaci.

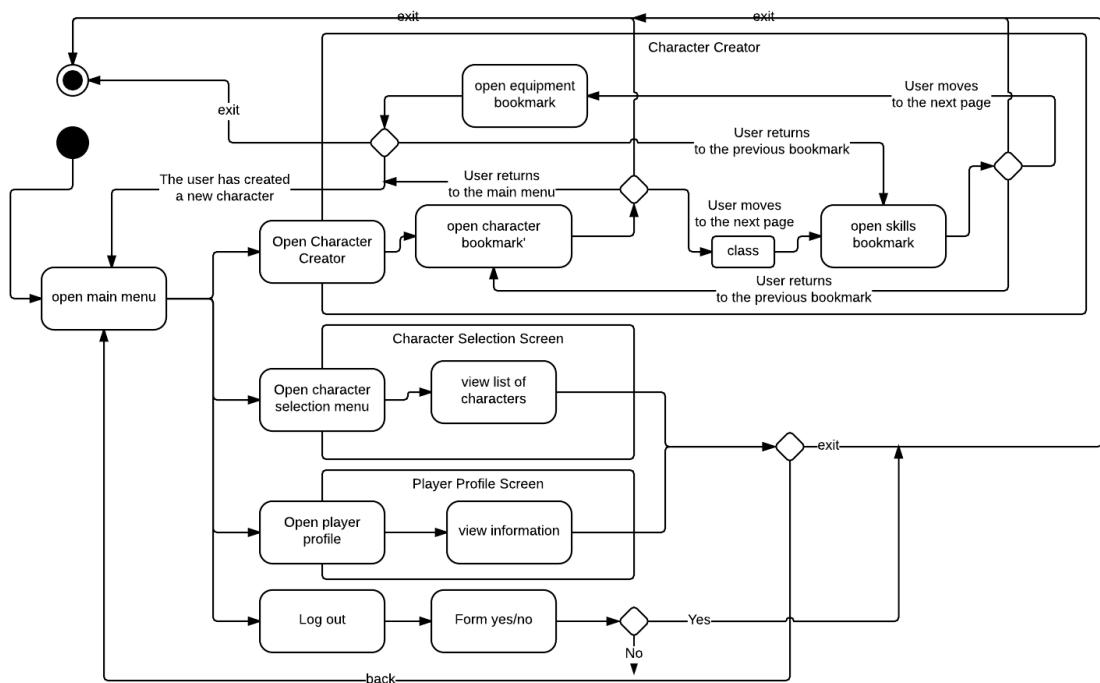
3.6 Diagram aktywności

3.6.1 Diagram aktywności - Dostępność do programu.



Rys.12 Diagram aktywności – Dostępność do programu(Logowanie i Rejestracja).

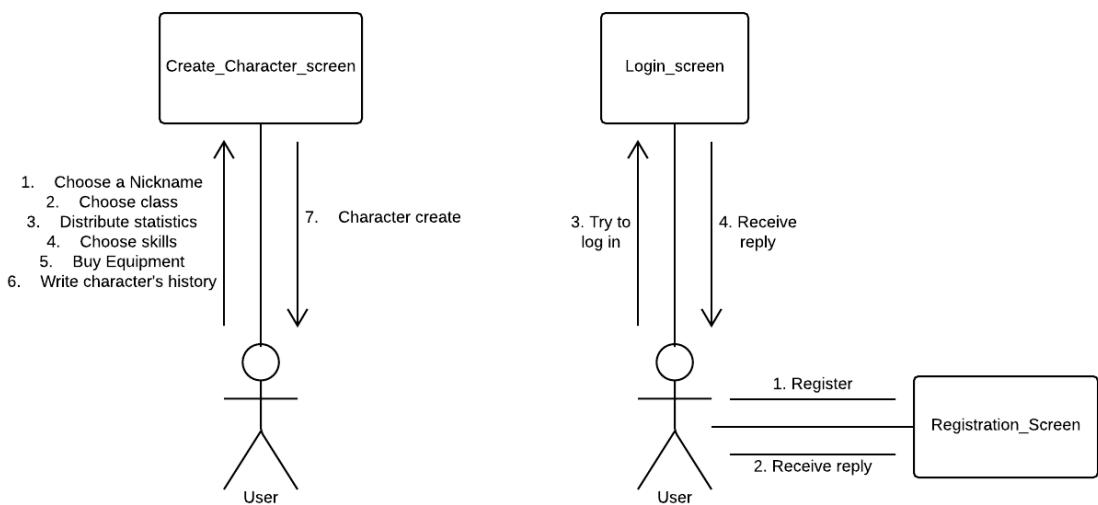
3.6.2 Diagram aktywności - Menu główne, kreator postaci, menu wyboru postaci, profil gracza.



Rys.13 Diagram aktywności – Menu główne i inne ekrany.

3.6 Diagram komunikacji/kooperacji.

Diagramy komunikacji/kooperacji mają na celu pokazać tutaj kolejność wykonywania działań przez użytkownika. W tym wypadku jasno widać, że użytkownik wprowadza dane, bądź wykonuje jakieś operacje w GUI, następnie te informacje zostają przetwarzane w innej części systemu, a interfejs wyświetla odpowiedź systemu.



Rys.14 Diagramy komunikacji/kooperacji

– Tworzenie postaci

Rys.15. Diagramy komunikacji/kooperacji

– Rejestracja/Logowanie

3.7 Krótkie podsumowanie.

Model interfejsu graficznego stawia przed nami dosyć spore wyzwanie przynajmniej na początku, gdy nie jesteśmy w stanie określić efektu końcowego innych części projektu. Jednak jako warstwa pośrednicząca między klientem, a systemem jest warta uważnego zaprojektowania, gdyż dzięki temu jako programiści jesteśmy w stanie pominąć najbardziej problematyczny etap. Etap w którym to posiadając całą funkcjonalność systemu, wielu programistów nie potrafi stworzyć wygodnego i przystosowanego dla każdego interfejsu graficznego.

Rozdział 5

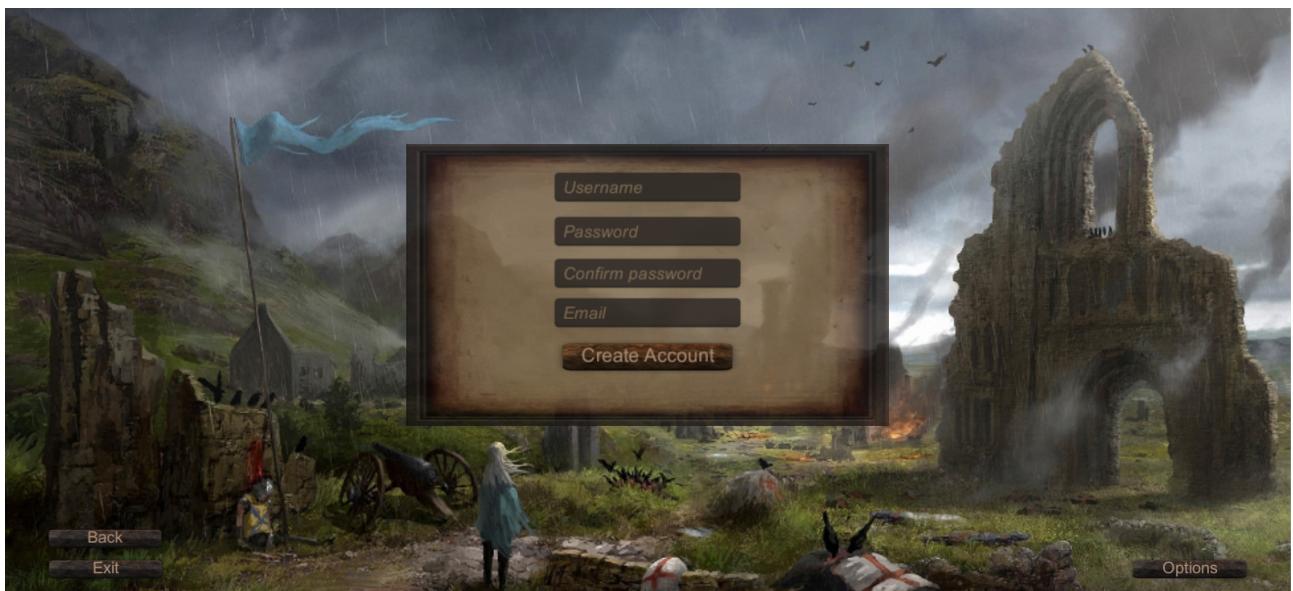
Screeny i opisy.

W tym rozdziale zaprezentuje efekt końcowy projektu w postaci screenów stworzone interfejsu użytkowego wraz z opisami.

5.1 Dostępność do programu

5.1.1 Rejestracja

Użytkownik, który chce mieć dostęp do gry musi się zarejestrować, jest to wymagane do użytkowania programu. Aby proces rejestracji zakonczył się powodzeniem, gracz musi uzupełnić formularz odpowiednimi danymi



Rys. * Ekran Rejestracji

Username – nazwa konta gracza, musi być unikalna, więc w razie zaistnienia już takowej w bazie danych, użytkownikowi zostanie zwrócony komunikat błędu

Password – Hasło, strzegące dostępu do konta gracza, dla potwierdzenia, że użytkownik się nie pomylił, musi powtórnie je wpisać w polu *Confirm password*.

Email – Adres email, wymagany, aby zapobiec tworzeniu przez gracza wielu kont. Też musi być unikalny, dla każdego gracza.

5.1.2 Logowanie

Użytkownik, który posiada już swoje konto, może dokonać procesu logowania i rozpocząć swoją przygodę.



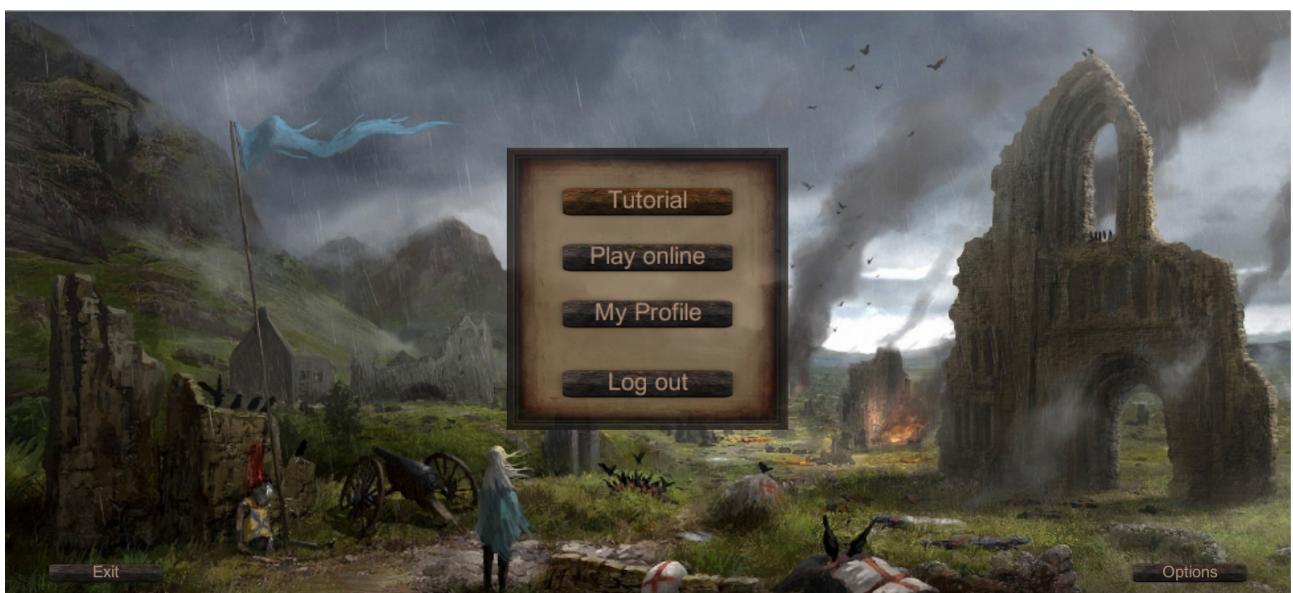
Rys. * Ekran Logowania

Email – wprowadzony uwcześnie przez użytkownika przy procesie rejestracji.

Password – wybrane uwcześnie przez użytkownika, ma zweryfikować, czy gracz loguje się na jego własne konto.

Registration – gdy użytkownik nie posiada jeszcze własnego konta, ten przycisk wysyła go do ekranu z formularzem rejestracji.

5.2 Menu Główne



Rys. * Ekran Menu Głównego

Tutorial – przycisk prowadzi do wprowadzenia dla nowych graczy.

Play online – przycisk prowadzi do menu z aktualnymi postaciami gracza, oraz gdzie może stworzyć nowe.

My Profile – otwiera stronę, gdzie user może zobaczyć informacje o jego profilu, oraz zmienić hasło.

Log out – wydelegowuje gracza z gry.

5.3 Profil gracza.

W tym oknie gracz może zobaczyć dane swojego konta, oraz je zmienić.



Rys. * Ekran profilu gracza.

Username – nazwa konta gracza, musi być unikalna, więc w razie zaistnienia już takowej w bazie danych, użytkownikowi zostanie zwrócony komunikat błędu

Email – Adres email, wymagany, aby zapobiec tworzeniu przez gracza wielu kont . Też musi być unikalny, dla każdego gracza.

Avatar – Ikona która będą widywać gracze np. w oknie kreatora walki.

5.4 Wybór Postaci.

Gracz ma do wyboru, może stworzyć nową postać, bądź ją wybrać z dostępnej listy.



Rys. * Ekran wyboru postaci.

Character Name – ten button przekierowuje gracza do okna kreatora postaci.

„Character's Name” - button z imieniem postaci (już istniejącej) rozpoczyna rozgrywkę właśnie tą postacią.

5.5 Tworzenie Postaci.

Zanim gracz rozpocznie swoją przygodę, musi stworzyć nową postać.



Rys. * Ekran tworzenia postaci.

Ikony klas – są tą przyciski wyboru klasy, sprawiają że formularz statysyk zostaje automatycznie uzupełniony.

Formukarz– kolejne elementy odpowiadają za statystyki postaci,

Character's Name – w tym miejscu gracz nadaje imię swojej postaci.

<< niedokończona część tworzenia postaci >>

Rysunki

Diagramy:

- Rys.1 Diagram przypadków użycia – GUI.
- Rys.2 Diagram klas dla ogólnej budowy ekranów w programie.
- Rys.3 Diagram klas dla elementów „Item”.
- Rys.4 Diagram klas dla elementów „Class_ch”.
- Rys.5 Diagram klas dla elementów „Dice”.
- Rys.5 Diagram sekwencji – Rejestracja.
- Rys.6 Diagram sekwencji – Login.
- Rys.7 Diagram sekwencji – Tworzenie walki PvP/TvT/GvG.
- Rys.8 Diagram sekwencji – Kreator postaci.
- Rys.9 Diagram stanów – Okna.
- Rys.10 Diagram stanów – Zakładki w oknie kreatora postaci.
- Rys.11 Diagram aktywności – Dostępność do programu(Logowanie i Rejestracja).
- Rys.12 Diagram aktywności – Menu główne i inne ekrany.
- Rys.13 Diagramy komunikacji/kooperacji – Tworzenie postaci.
- Rys.14 Diagramy komunikacji/kooperacji – Rejestracja/Logowanie.

Screeny:

- Rys. * Ekran Rejestracji.*
- Rys. * Ekran Logowania.*
- Rys. * Ekran Menu Głównego*
- Rys. * Ekran profilu gracza.*
- Rys. * Ekran wyboru postaci.*
- Rys. * Ekran tworzenia postaci.*

Tabele:

Tab. 1 Tabela do Diagramu przypadków użycia (Rys. 1).

Bibliografia

Dokumentacja Unity, <http://docs.unity3d.com/Manual/GUIScriptingGuide.html>

Zosia Kruczkiewicz, Wykład,

<http://zofia.kruczkiewicz.staff.iiar.pwr.wroc.pl/wyklady/analizasi/interfejsuzytkownika.pdf>

B. Bruegge, A. H. Dutoit, *Inżynieria oprogramowania w ujęciu obiektowym. UML, wzorce projektowe i Java.* , Helion, 2011.