Uniwersytet Jagielloński

Wydział Informatyki i Matematyki

INSTYTUT INFORMATYKI I MATEMATYKI KOMPUTEROWEJ

Studia stacjonarne

Nr albumu: 1088807 Ocena:

Damian Misiura

Teiru MMORTS   
(Massively multiplayer online real-time strategy)  
Projekt i implementacja części klienckiej

Opiekun pracy licencjackiej:

Dr Marcin Żelawski

Opracowano zgodnie z ustawą z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz.U. 1994 nr 24 poz. 83 ze zm. Tekst jedn.: Dz.U 2006 nr 90 poz. 631 ze zm.)

Kraków 2015

**Abstrakt**

Celem tej pracy będzie ukazanie budowy aplikacji klienckiej do gry typu massively multiplayer real-time strategy, czyli do gry wieloosobowej strategicznej czasu rzeczywistego Teiru MMORTS. Aplikacja kliencka jest niezbędna do rozpoczęcia rozgrywki oraz komunikacji z serwerem gry. Dzięki niej użytkownik może rozpocząć interakcję z innymi graczami tejże gry, walczyć z nimi, tworzyć nowe zorganizowane grupy. Każdy gracz ma możliwość tworzenia nowych elementów tej gry, na przykład mapy bądź przedmiotów. Gra oferuje rozwój postaci, wykonywanie ciekawych zadań, podróż po świecie gry. Aplikacja kliencka powinna realizować wyświetlanie wszystkich z powyżej wymienionych funkcji. Musi też zagwarantować stabilność połączenia z serwerem oraz przeprowadzenie rozgrywki bez problemów.

The goal of this work is to show the construction of the client application to game named Teiru MMORTS, which type of massively multiplayer real-time strategy. The client application is necessary to start the game and communication with game server. Thanks to it, user can start interaction with another players of this game, fight with them, create new organized groups. Every player has the ability to create new elements of this game, such as new maps or new items. The game itself offers character development, interesting tasks to perform, ability to make journey through the world of the game. Client application should pursue to display all of the above functions. It has to ensure stability of connection with server and perform the game without problems.

Spis treści

[1. Część teoretyczna UML 5](#_Toc408932336)

[1.1. Czym jest UML? 5](#_Toc408932337)

[1.2. Praca w UML 5](#_Toc408932338)

[1.2.1. Diagram przypadków użycia oraz scenariusze przypadków użycia 5](#_Toc408932339)

[1.2.2. Diagram stanów 6](#_Toc408932340)

[1.2.3. Diagram klas 6](#_Toc408932341)

[1.2.4. Diagram czynności 6](#_Toc408932342)

[1.2.5. Diagram sekwencji 6](#_Toc408932343)

[1.2.6. Diagram komunikacji 7](#_Toc408932344)

[1.2.7. Diagram komponentów 7](#_Toc408932345)

[1.2.8. Diagram wdrożenia 7](#_Toc408932346)

[2. Funkcjonalna charakterystyka systemu 8](#_Toc408932347)

[2.1. Charakterystyka gry 8](#_Toc408932348)

[6. Szczegóły techniczne dotyczące implementacji 9](#_Toc408932349)

[7. Bibliografia 10](#_Toc408932350)

**Wstęp**

Tematem pracy jest stworzenie aplikacji klienckiej dla gry typu MMORTS, czyli gry strategicznej czasu rzeczywistego dla wielu graczy.

Aplikacja ta powinna udostępniać użytkownikowi rozgrywkę w czasie rzeczywistym wraz z innymi graczami. Gracze powinni mieć możliwość odczucia dynamiki gry oraz uczucia ciągle rozwijającej się akcji. Aplikacja powinna udostępnić użytkownikowi możliwość tworzenia nowych map bądź nowych przedmiotów.

Gry MMORTS ostatnimi latami przyciągają ogromną ilość graczy z powodu dynamicznej rozgrywki, trudności polegającej na szybkości podejmowanych decyzji bądź rywalizacji pomiędzy nimi.

Celem tej pracy jest ukazanie procesu budowy tejże aplikacji, schematu jej działania oraz efektów jej pracy. Aplikacja jest budowana na bazie pakietu narzędzi Unity oraz technologii .NET. Wykorzystywane w niej są algorytmy bezpiecznego połączenia z serwerem oraz algorytmy typu znajdowanie najkrótszej drogi bądź optymalnego zarządzania obiektami w grze.

Pierwszy rozdział opisywać będzie czym jest język formalny modelowania UML oraz z jakich części jest zbudowany.

Drugi rozdział

Trzeci rozdział będzie opisywać budowę modelów aplikacji klienckiej, ich struktury oraz celu zastosowania odpowiednich modelów.

Czwarty rozdział opisywać będzie część logiczną aplikacji, strukturę modułów zbudowanych w tejże apliacji oraz ich wykorzystanie.

Piąty rozdział będzie pokazywał proces implementacji aplikacji klienckiej, ukazane będą fragmenty aplikacji które będą miały kluczowe znaczenid o działania całego systemu. (semestr 2)

Szósty rozdział opisywał będzie użyte zewnętrzne materiały do budowy aplikacji, użyte algorytmy, grafiki bądź frameworki oraz wymagania systemowe gry. (częściowo semestr 2)

Siódmy rozdział opisywał będzie zewnętrzne książki, artykuły bądź prace naukowe z który zaczerpana została wiedza do napisana tej pracy.

# Część teoretyczna UML

## Czym jest UML?

UML (ang. Unified Modeling Language) [1] jest ustandaryzowaną notacją modelowania obiektów w świecie rzeczywistym, nazywany jest też językiem formalnym do modelowania tychże obiektów. Notacja została stworzona poprzez złączenie trzech notacji opartych o obiektowość oraz metodologii analizy:

* Metodologia Grady Booch’a do opisywania zbioru obiektów i zależności między nimi
* Podejście Ivara Jacobson’a które zawierające też metodę przypadków użycia
* Object-Modeling Technique Jamesa Rumbaugh’a

Tworzenie projektu w oparciu o język UML jest pierwszym krokiem do tworzenia projektów opartych o projektowanie oparte o metodykę obiektową.

Inne pomysły również zostały włączone to standardu, co było zasługą Boocha, Rumbaugha, Jacobsona oraz innych osób pracujących dzięki sponsorowaniu przez Rational Software. UML został dopracowany i zaakceptowany przez Object Management Group (OMG). Obecnie UML jest wykorzystywany przez prawie każdego największego dostawcę oprogramowania na świecie, wliczając w to IBM lub Microsoft.

## Praca w UML

W notacji UML modele bądź obiekty przedstawiane są jako różnego rodzaju diagramy. Reprezentowane są one przez obiekty które przedstawione są jako kształty bądź obrazki zawierające odpowiednią treść. Wyróżnia się wiele rodzajów diagramów [2]:

### Diagram przypadków użycia oraz scenariusze przypadków użycia

Diagram przypadków użycia opisuje zachowanie systemu z punktu widzenia użytkownika. Dla deweloperów oprogramowania ten typ diagramów okazuje się być bardzo przydatnym narzędziem, dzięki niemu metodą prób i błędów można zobaczyć założenia użytkowników wobec tworzonego systemu. Diagramy te służą ukazaniu tego co chce użytkownik wobec systemu.

Scenariusz przypadków użycia pokazuje oczekiwanie użytkownika względem konkretnie wykonywanej akcji, proces jej wykonywania oraz etapy w których owa akcja jest wykonywana. Przykładem jest proces kupna jachtu.

### Diagram stanów

Diagram stanów opisuję zmianę stanu obiektów względem wykonywanej akcji. Z wykonaniem każdego etapu tej akcji zmienia się stan obiektu bądź modułu systemu, ich zmiany pokazane są na diagramie. Dla programistów jest to ważny diagram ponieważ mogą zauważyć zmianę zachowania obiektu wobec wykonywanej czynności na nim.

### Diagram klas

Diagram klas opisuje wszystkie obiekty tworzone w obecnym module, ich powiązania, na przykład zawieranie się w sobie bądź związki dziedziczenia. Obiekty mają określone typy, funkcje oraz metody. Każdy element modułu przedstawiany na diagramie ma określoną formę, nie opisywane są konkretne obiekty ale jedynie ich abstrakcyjne formy. Dla programisty jest to szczególnie ważne ponieważ diagram ten pokazuje mu jak zbudowany jest system i jakie są w nim zależności. Również dzięki niemu programista jest w stanie łatwo zaimplementować te powiązania między obiektami.

### Diagram czynności

Diagram czynności pokazuje jak zachowuje się system w odniesieniu do obecnie wykonującej się operacji.   
Jest bardzo podobny do diagramu stanów, jednak zamiast opisywać zachowanie jednego obiektu opisuje zachowania wielu obiektów, między którymi nawiązywana jest komunikacja. Pokazuje on programistom sposób działania pewnych czynności, ułatwiając implementację opisywanej diagramem czynności.

### Diagram sekwencji

Diagram sekwencji prezentuje kolejność wykonania etapów pewnej akcji, przepływ sterowania informacjami pomiędzy modułami systemu oraz szablon opisywanego algorytmu. Nie jest uwzględniany dostęp do danych, operacje na danych oraz innych funkcji związanych z komunikacją.

### Diagram komunikacji

Diagram skupia się na obiektach które wchodzą w skład interakcji oraz komunikatach wymienianych przez nich. Aspekt czasowy ma małe znaczenie, z tego powodu umieszczenie obiektów na diagramie ma cel łatwego opisania relacji między nimi. Diagram opisuje w jaki sposób komponenty komunikują się ze sobą podczas wykonywania danej akcji.

### Diagram komponentów

Diagram komponentów pokazuje podział systemu na mniejsze podsystemy logiczne. Podsystemy te są połączone między sobą poprzez wspólne funkcje bądź wspólne zapotrzebowanie na daną porcję danych. Komponent to fizyczny moduł kodu, zwykle pakiet. Musi on posiadać swoją unikalną nazwę. Można tworzyć grupy komponentów zwane pakietami, które mogą zostać podsystemami.

### Diagram wdrożenia

Diagram wdrożeń pokazuje powiązania pomiędzy oprogramowaniem a sprzętem fizycznym. Wykorzystywane są przy modelowaniu dużych systemów informatycznych.

# Funkcjonalna charakterystyka systemu

System Teiru MMORTS będzie składał się z trzech części:

* Aplikacji klienckiej
* Serwera gry
* Bazy danych

Wyżej wymieniona baza danych będzie podzielona między serwer a klienta. W tym rozdziale bardziej skupię się na aplikacji klienckiej będącej tematem tej pracy, jednak postaram się opisać zasadę działania całego systemu. System wymaga rejestracji aby podjąć w nim jakąkolwiek decyzję.

## Charakterystyka gry

Budowany system jest grą strategiczną czasu rzeczywistego skierowaną do rozgrywek dla dużej ilości graczy. Sama gra będzie oferować tryb pojedynczego gracza (single player mode) oraz tryb wielu graczy (multi player mode). Tryb pojedynczego gracza ma za zadanie zapoznać nowego z założenia gracza z zasadami gry, pokazać mu podstawowe mechanizmy gry oraz przygotowanie do rozgrywek trybu multi player. Sam tryb multi player umożliwia stworzenie nowej postaci, dostosowanie jej statystyk do własnych potrzeb.

Zalogowanie się do systemu jest warunkiem koniecznym do rozpoczęcia jakiegokolwiek trybu rozgrywki, tworzenia postaci bądź rozmowy z innymi graczami.

Rozgrywka odbywa się na zasadzie rund, gdzie każdy ruch gracza bazowany jest na rzucie kostką. Rundy są ograniczone czasowo aby zachować płynność rozgrywki oraz zapewnić brak celowego wstrzymywania rozgrywki. W każdej rundzie gracz może przesuwać się po mapie, prowadzić interakcje z innymi graczami, statycznymi postaciami na mapie, wykonywać zadania, prowadzić walki.

Każdy gracz ma możliwość stworzenia gildii, to znaczy grupy osób zrzeszonej w tej samej grupie, które mają możliwość rozmowy w prywatnym kanale bądź braniu udziału w walkach pomiędzy innymi gildiami. Gracze mają również tworzenia listy przyjaciół w grze, dzięki której możliwe będzie sprawdzenie obecnego stanu przyjaciela lub prywatnej rozmowy z nim.

Postać stworzona przez gracza może zostać wybrana z jednej z czterech dostępnych klas. Są to Ranger, Mage, Knight oraz Mystiq. Każda klasa postaci ma swoje zalety jak i wady, każda też wyróżnia się stylem walki. Przykładowo Mage włada magią w celach ofensywnych, Knight z drugiej strony zawiera dużą ilość obrony i walczy wręcz. Postaci z czasem mogą awansować na wyższe poziomy, aby odblokować nowe zaklęcia, umiejętności oraz rozdzielić nowo dobyte punkty statystyk. Statystyk w grze jest sześć, są to:

* Strength – zwiększa możliwy ładunek gracza oraz ilość obrażeń zadawanych w walce wręcz
* Dexterity – określa skuteczność ataków dystansowych oraz ich obrażenia
* Constitution – wytrzymałość, określa ilość punktów życia postaci
* Intelligence – zwiększa ilość punktów statystyk i umiejętności przyznawanych na każdy nowy poziom
* Wisdom – mądrość, zwiększa siłę zaklęć
* Charisma - zwiększa odporność na magiczne obrażenia

Gracz pierwszopoziomowy otrzymuje wartość 8 każdej ze statystyk, aby zwiększyć każdą z  
nich należy rozdysponować punkty statystyk otrzymywane za każdy nowy poziom. Kolejno 1 punkt statystyki od 9 do 14, po 14 punkcie statystyk potrzebne są 2 punkty aby awansować. Co każdy poziom postać ma możliwość wyboru nowych zaklęć bądź indywidualnych umiejętności, wyspecyfikowanych dla każdej klasy.

Postać ma możliwość tworzenia ekwipunku, zarządzania nim, zbierania nowych przedmiotów z zabitych potworów, nakładania nowych elementów ekwipunku, kupna nowych przedmiotów oraz sprzedawania niepotrzebnych przedmiotów.

Gra posiada czat, w którym wszyscy gracze mogą rozmawiać ze sobą. Czat składa się z podczatów: czatu gry, czatu ogłoszeń bądź czatu logu. Gracze mogą tworzyć własne kanały do rozmowy poprzez odpowiednią akcję w grze. Gildie również posiadają swój odrębny czat, tak samo jaki walki.

Walki w grze dzielą się na trzy rodzaje:

* PvE – gracz kontra środowisko, gdzie walka odbywa się pomiędzy stworzenia ze świata gry
* PvP – gracz kontra gracz, pojedynek przeciwko innemu graczowi
* GvG – gildia kontra gildia, walka 4 graczy na 4 graczy zwaśnionych gildii

Walki PvE oznaczają walkę przeciw stworzeń ze świata gry, walka ta może być przykładowo wyznaczona przez zadanie, rozpoczęta losowo bądź znaleziona poprzez losowe wydarzenie na mapie. Nagroda za nie jest duża ilość doświadczenia, losowa ilość przedmiotów bądź nowe elementy ekwipunku.

Walki PvP oznaczają pojedynki między graczami, odbywają się one na zasadzie jeden na jednego. Nagrodą za nie może być nowa pozycja w globalnym rankingu, złoto, przedmioty bądź jakiś element ekwipunku przeciwnika oraz wieczna chwała.

Walki GvG określają walkę pomiędzy dwiema gildiami. Toczą się na zasadzie 4 graczy na 4 graczy, przy czym nie ma możliwości aby walki posiadały mniej lub więcej graczy. Nagrodą za walkę jest złoto, przedmioty i ekwipunek podzielony na zwycięska drużynę, nowa pozycja rankingowa gildii.

Sama walka odbywa się turowo, początkowo tworzona jest kolejka postaci wykonujących ruch względem ich inicjatywy. Następnie kolejka ruchów cyklicznie przesuwa się do momentu aż postaci jednej frakcji zostaną w tej kolejce eliminując drugą stronę.

Walka zawiera typy akcji, a więc: akcje standardową, czyli np. atak bądź ruch po mapie. Akcja ruchu oznacza poruszanie się po mapie przez postać, może być więc akcją standardową. Akcja całorundowa to skupienie całej tury postaci na jednym zadaniu zamiast przykładowo na ruchu i następującym ataku.

Podczas ataku wykonywany jest test ataku aby sprawdzić czy próba ataku się powiodła, oceniane jest to przez rzut kostką K20. Jeżeli atak się powiedzie (czyli rzut kostką będzie korzystny) następuje obliczenie otrzymanych obrażeń przez cel, kostka tutaj również ma wpływ. Minimalne obrażenia zawsze wynoszą 1, maksymalne są obliczane przez modyfikatory typu umiejętności, statystyk bądź broni lub zaklęcia. Postaci mogą być w różnych stanach w zależności od otrzymanych obrażeń, np. Zdrowy, Okaleczony, Martwy.

Rozgrywka całej gry oparta jest głównie na walkach, dzięki czemu gracz ma możliwość zaobserwowania dynamiki całej akcji mającej miejsce w grze.

## Charakterystyka aplikacji klienckiej

# 6. Szczegóły techniczne dotyczące implementacji

Wykorzystane technologie w budowie aplikacji: C#, .NET, pakiet Unity 4.7.1, MySQL, DirectX 9c.

*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

# 7. Bibliografia

[1] Wikipedia, UML, <http://pl.wikipedia.org/wiki/Unified_Modeling_Language>

[2] Joseph Schmuller, „UML dla każdego”, tłumaczenie Krzysztof Masłowskiego, Gliwice 2003

Disclaimer:

Poniższa praca jest w trakcie edycji, prawie każde zdanie, diagram bądź inna forma informacji może być zmieniona w trakcie tworzenia projektu.