Licencjacka Pracownia Oprogramowania, zespół 12
Instytut Informatyki Uniwersytetu Wrocławskiego

Wtomigraj

Architektura projektu

Marcin Januszkiewicz, Grzegorz Łoś

Wrocław, 29 czerwca 2012

Spis treści

1. Ogólny opis architektury

Każdy klient może zostać gospodarzem nowej gry lub przyłączyć się do nierozpoczętej gry (mówimy wtedy, że jest gościem). Kiedy do nierozpoczętej partii dołączy minimalna liczba wymaganych graczy (zależy ona od gry), to partia może być rozpoczęta. Serwer służy tylko jako pośrednik w nawiązaniu połączenia pomiędzy graczami zainteresowanymi odbyciem wspólnej partii, nie ma żadnego udziału w jej przebiegu. Wszyscy goście komunikują się wyłącznie z gospodarzem. Wszystkie obiekty związane z grą znajdują się w pamięci komputera gospodarza. Klient gospodarza wysyła gościom informacje o zmianach stanów tych obiektów, a klienci-goście wyświetlają je swoim użytkownikom na ekranach monitorów i przekazują gospodarzowi ich ruchy. Po zakończeniu gry wszyscy klienci ponownie komunikują się z serwerem, który może pośredniczyć w zainicjowaniu kolejnej partii. Schemat architektury przedestawiono na rysunku 1.

Końcowymi użytkownikami są internauci pragnący pograć z innymi. Programista gry korzystający z platformy Wtomigraj powinien na wydzielonym serwerze uruchomić program serwera oraz udostępnić użytkownikom grę, na przykład w postaci aplikacji okienkowej w pakiecie jar lub jako aplet osadzony na stronie internetowej.

1.1. Stany partii

Partia może znajdować się w jednym z trzech stanów:

1. **Nierozpoczęta.** W tym stanie do partii mogą dołączać gracze. Nazwa partii jest widoczna w aplikacjach klientów znajdujących się w trybie menu. Gospodarz partii może ustawiać parametry związane z grą.

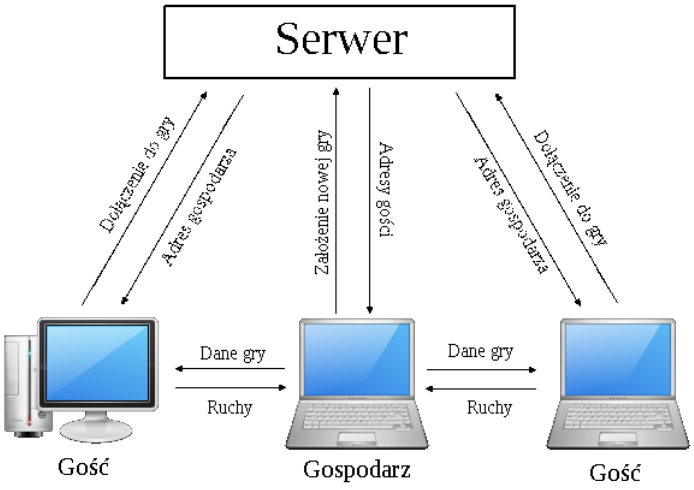
2. **Rozpoczęta.** Nazwa partii znika z listy widocznej na rysunku 2, użytkownicy tracą możliwość dołączania do niej. Gospodarz otrzymuje od serwera adresy gości, a goście adres gospodarza. Na komputerze gospodarza inicjowane są wszystkie obiekty związane z grą i partia się rozpoczyna.

3. **Zakończona.** Partia się zakończyła, lecz gospodarz jeszcze jej nie opuścił. W tym stanie gracze mogą oglądać wyniki, analizować statystyki gry lub dyskutować nad przebiegiem partii.

1.2. Tryby klienta

Aplikacja klienta w każdym momencie znajduje się w jednym z trzech trybów.

Tryb menu. Jest to tryb między partiami.



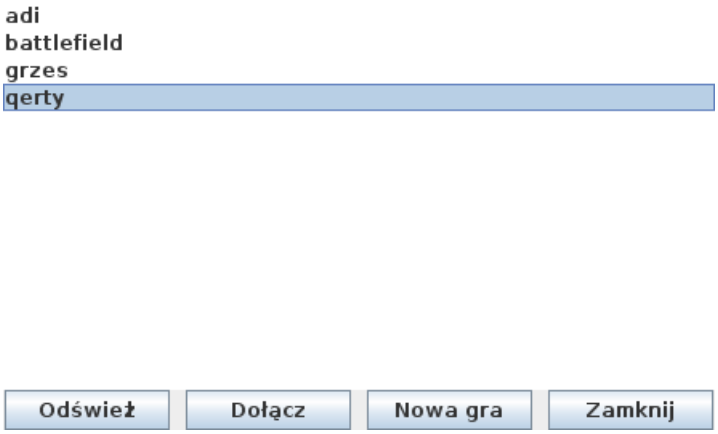
Rysunek 1. Schemat komunikacji klientów i serwera

Aplikacja prezentuje swojemu użytkownikowi partie, do których może się przyłączyć, daje możliwość założenia nowej partii oraz porozmawiania z pozostałymi graczami. Przedstawiono to na rysunku 2.

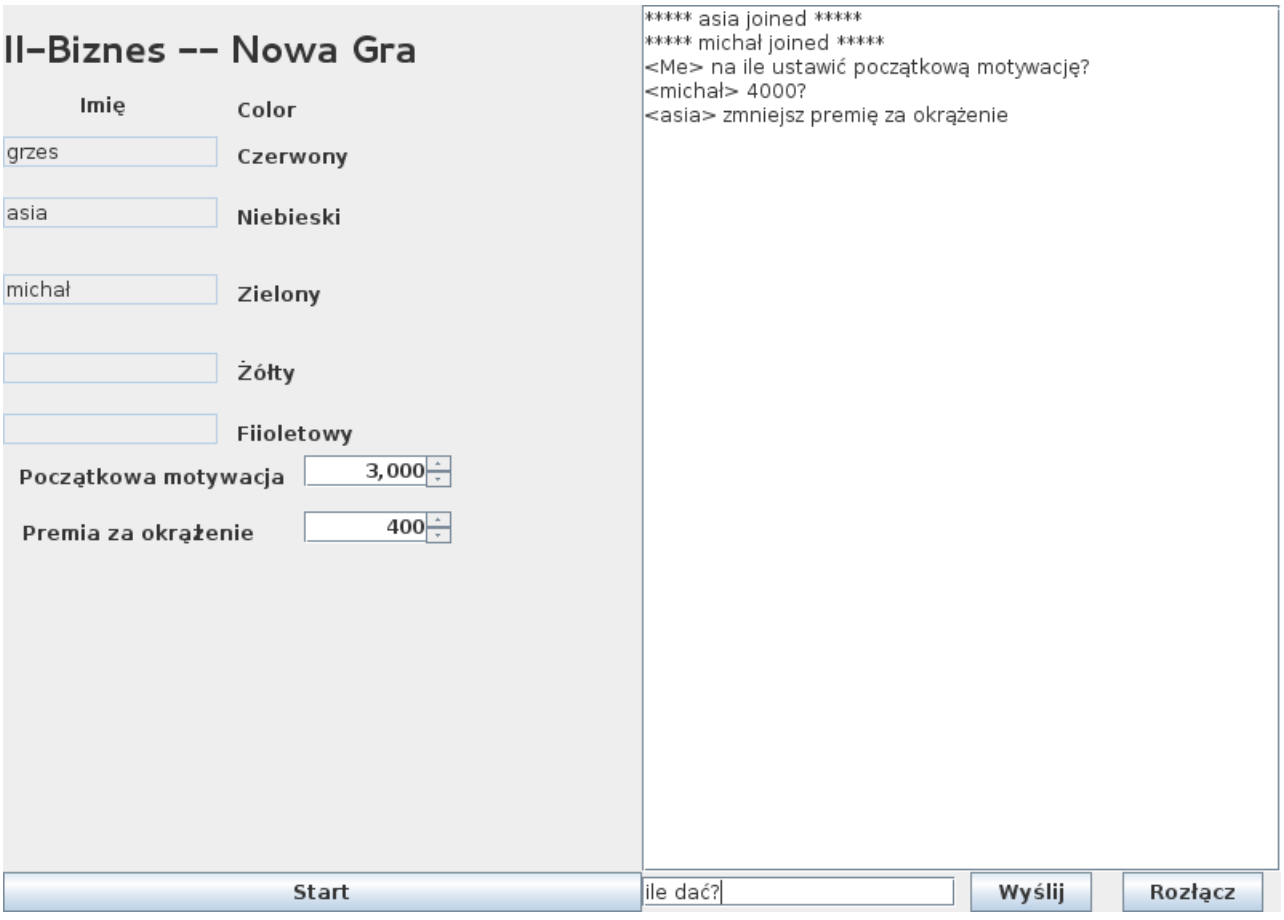
Tryb gospodarza. W tym trybie klient staje się gospodarzem partii. Początkowo oczekuje aż dołączy odpowiednio wielu graczy. Użytkownicy, których klienci znajdują się w trybie menu, widzą tę partię na swoim panelu i mogą do niej dołączyć. W tym czasie gospodarz może ustawić parametry związane z grą, co przedstawia rysunek 3.

Po pojawieniu się wymaganej liczby graczy partia może być rozpoczęta. Znika wówczas z listy dostępnych gier. Aplikacja gospodarza odpowiada za wykonywanie wszelkich obliczeń i operacji związanych z grą. Wszystkich gości informuje tylko o zmianach stanów istotnych obiektów, tak aby aplikacje gości mogły je należycie wyświetlić swoim użytkownikom. Goście przekazują swoje ruchy gospodarzowi, który uwzględnia je w logice gry. Ponadto aplikacja w trybie gospodarza wykonuje te same zadania, co aplikacja w trybie gościa, ponieważ jej użytkownik także jest graczem.

Tryb gościa. W tym trybie aplikacja służy do wyświetlania stanu gry swojemu użytkownikowi oraz przekazywania jego ruchów gospodarzowi partii.



Rysunek 2. Aplikacja w stanie menu



Rysunek 3. Przykładowy panel gospodarza przed rozpoczęciem partii

2. Zadania elementów systemu

Serwer. Służy skomunikowaniu aplikacji klientów. Może obsługiwać dowolnie wiele rodzajów gier. Serwer:

- przechowuje adresy klientów, ich pseudonimy, oraz odnośniki do partii w jakich się znajdują,
- rozsyła listę dostępnych partii pożądanej gry (aplikacja klienta wysyła informację do serwera jaka gra ją interesuje, bo serwer może jednocześnie służyć np. szachistom i brydżystom),
- pośredniczy w nawiązaniu komunikacji przez gospodarza i gościa.

Aplikacja klienta. Jest implementacją pewnej gry, korzystającą ze stworzonej przez nas biblioteki do komunikacji.

1. W trybie menu:
   - umożliwia tekstową komunikację użytkowników w okienku czata,
   - przedstawia listę nierozpoczętych partii,
   - daje możliwość dołączenia do nierozpoczętej partii lub założenia nowej.
2. W trybie gospodarza:
   - przechowuje adresy klientów gości,
   - trzyma w pamięci wszystkie obiekty związane z partią,
   - służy ustawieniu parametrów partii i jej rozpoczęciu,
   - odpowiada za uaktualnianie logiki gry,
   - przyjmuje od gości informacje o ich ruchach,
   - wysyła gościom informacje o aktualnym stanie partii,
   - pełni także funkcje klienta w trybie gościa, ponieważ gospodarz także jest graczem.
3. W trybie gościa:
   - przechowuje adres gospodarza,
   - przyjmuje od gospodarza komunikaty o aktualnym stanie partii,
   - wyświetla użytkownikowi stan gry (na przykład w przypadku szachów będzie do rysunek szachownicy i bierek),
   - oczekuje na ruchy swojego użytkownika (w przypadku szachów kliknięcia na bierki i pola), a następnie wysyła je gospodarzowi.

3. Opis protokołu

Komunikacja pomiędzy klientem a serwerem odywa się przez wymianę pakietów UDP z wiadomościami zawartymi w polach danych. W warstwie aplikacji protokół ma formę pytanie-odpowiedź, co zapewnia odporność na błędy sieci związane ze stratą pakietów. Wiadomości przesyłane są w formacie JSON, co pozwala na wykorzystanie istniejących bibliotek do prostego kodowania i dekodowania informacji.

Każdy obiekt JSON zawiera pola:

- `type` – typ komunikatu, łańcuch znaków. Wszystkie typy komunikatów omawiamy poniżej,
- `res` – pole boolowskie ustawione na true, gdy pakiet jest odpowiedzią, w przeciwnym razie jest to pytanie,
- `id` – w przypadku pakietów-pytań jest to numer identyfikacyjny pytania, natomiast dla odpowiedzi oznacza identyfikator pytania, na które odpowiadamy.

Ponadto dla każdego typu komunikatu obiekt JSON może zawierać dodatkowe pola.

Inne pola. Opisane poniżej pola występują w niektórych typach komunikatów.

- `nick` – pseudonim używany przez klienta.
- `name` – nazwa partii, do której odnosi się komunikat.
- `game` – nazwa (rodzaj) gry, do której odnosi się komunikat.
- `channels` – lista nazw partii, do których może dołączyć klient.
- `desc` – słowny opis komunikatu.

3.1. Typy komunikatów

Komunikaty wysyłane przez klienta do serwera.

newclient – komunikat informujący o chęci przyłączenia się do serwera. Zawiera pole `nick` (pożądany pseudonim).

newchannel – komunikat będący prośbą o założenie nowej partii. Zawiera pola `name` oraz `game`.

join – komunikat będący prośbą o przyłączenie do partii. Zawiera pola `name` oraz `game`.

exit – komunikat będący informacją o odłączeniu od partii.

sendchannels – komunikat będący prośbą o przesłanie listy partii do których może dołączyć klient. Zawiera pole `game`.

emptyresponse – komunikat wysyłany w odpowiedzi na pytania, które *de facto* są poleceniami i nie wymagają żadnej odpowiedzi, prócz potwierdzenia, że pakiet doszedł, np. **echorequest**, **channelcanceled**.

Komunikaty wysyłane przez serwera do klienta.

welcome – odpowiedź na **newclient**. Komunikat ten oznacza, że serwer zaakceptował wybrany identyfikator. Dla potwierdzenia znajduje się on w polu `nick`. Ponadto pakiet zawiera pole `channels`.

invalidnick – odpowiedź na **newclient**. Komunikat ten oznacza, że wybrany identyfikator nie został zaakceptowany. Zawiera pola `errid` oraz `desc` oznaczające numer błędu oraz jego opis.

echorequest – komunikat żądający od klienta potwierdzenia swojej obecności.

channellist – odpowiedź na **sendchannels**. Zawiera pole `channels`.

channelaccepted – odpowiedź na **newchannel**. Komunikat ten oznacza, że serwer akceptuje założenie przez klienta nowego kanału rozmowy.

channelrejected – odpowiedź na **newchannel**. Komunikat ten oznacza, że serwer odrzuca założenie przez klienta nowego kanału rozmowy. W treści komunikatu przesyłany jest numer błędu i jego opis.

joinaccepted – odpowiedź na **join**. Komunikat ten oznacza, że prośba klienta o dołączenie do pewnego kanału została zaakceptowana. W polu `name`, jako potwierdzenie, znajduje się nazwa kanału, do którego dołącza klient.

joinrejected – odpowiedź na **join**. Komunikat ten oznacza, że prośba klienta o dołączenie do pewnego kanału została odrzucona.Zawiera pola `errid` oraz `desc` oznaczające numer błędu oraz jego opis.

exitaccepted – odpowiedź na **exit**. Oznacza on, że klient został wypisany z kanału rozmowy. Ponadto w polu `channels` zawarta jest lista nazw kanałów, do których może dołączyć klient.

address – Pakiet zawiera pola `address` oraz `port` będące adresem pewnego klienta.

channelcanceled – komunikat informujący klienta, że partia w której się znajduje przestał istnieć.

userleft – komunikat informujący gospodarza, że partię w którym się znajduje opuścił użytkownik. W treści komunikatu przesyłany jest identyfikator odchodzącego użytkownika.

error – komunikat wysyłany gdy wystąpi błąd inny niż te opisane wyżej. Zawiera pola `errid` oraz `desc` oznaczające numer błędu oraz jego opis.

Komunikaty wysyłane prez klienta do klienta.

holepunch – Pakiet, który ma na celu „wybicie dziury" (opisane w dziale 4.).

icanhearyou – Odpowiedź na **holepunch**.

gamedata – pakiet zawierający dane gry. Pozostałe pola pakietu są zdefiniowane przez programistę gry.

4. Komunikacja między serwerem i klientami

4.1. Komunikaty z serwera do klienta

Tabela 1 zawiera słowny opis komunikatów wysyłanych z serwera do klienta i pożądane zachowanie aplikacji klienta.

4.2. Komunikaty od klienta do serwera

Tabela 2 zawiera słowny opis komunikatów wysyłanych od klienta do serwera i zachowanie serwera.

4.3. Komunikacja gość-gospodarz

Nawiązywanie połączenia między klientami. W celu skomunikowania dwóch klientów korzystamy z metody „wybijania dziur" (ang. *holepunching*). Polega ona na tym, by do komputera, z którym chcemy się skontaktować wysyłać jakiekolwiek pakiety (ich zawartość jest nieważna). Robimy tak ze względu na to, że rutery często odrzucają pakiety od nieznanego nadawcy. Jednak kiedy ruter zobaczy, że otrzymujemy pakiet z adresu, do którego my także już coś wysyłaliśmy, to adres ten uzna za zaufany i pakiety zaczną przechodzić.

Komunikacja między gospodarzem i jego gośćmi w trakcie partii może być dowolnie zdefiniowana przez programistę gry.

Tabela 1. Komunikaty z serwera do klienta.

Tryb klienta	Rodzaj komunikatu	Co powinien zrobić klient
Menu	Wiadomość	Wyświetlić ją w stosownym okienku
	Lista dostępnych partii	Wypisać je na przeznaczonej do tego liście
	Akceptacja utworzenia nowej partii	Przełączyć się w tryb gospodarza, wyświetlić panel z ustawieniami partii.
	Odrzucenie nowej partii	Wyświetlić komunikat z przyczyną
	Akceptacja dołączenia do partii	Przejście w tryb gościa
	Odrzucenie prośby o dołączenie do partii	Wyświetlić komunikat z przyczyną
Gospodarz	Dołączenie gracza	Sprawdzić czy można rozpocząć partię
	Odejście gracza	Jeżeli partia nie jest rozpoczęta, to być może należy wyłączyć możliwość rozpoczęcia gry. Jeżeli jest rozpoczęta, to postępowanie zależy od konkretnej gry
Gość	Odejście gospodarza	Wyświetlić komunikat o przerwaniu gry i wrócić do trybu menu

Tabela 2. Komunikaty od klienta do serwera.

Tryb klienta	Rodzaj komunikatu	Co powinien zrobić serwer
Menu	Utworzenie nowej partii	Jeżeli nazwa gry jest poprawna, to: • poinformować klienta o akceptacji partii, • przesłać pozostałym klientom komunikat o nowej partii, • utworzyć obiekt partii, • przepisać klienta z listy wolnych do zajętych z referencją do utworzonej partii. W przeciwnym razie wysłać klientowi komunikat o odrzuceniu partii
	Prośba o dołączenie do partii	Jeżeli z jakiegoś powodu klient nie może do niej dołączyć, to wysyłamy stosowny komunikat. W przeciwnym razie tworzymy odnośnik do partii, w której znajduje się klient
Gospodarz	Odejście z partii	Wszystkim gościom wysyłamy komunikat o odejściu gospodarza. Przepisujemy wszystkich z listy zajętych do wolnych i usuwamy obiekt partii.
	Rozpoczęcie partii	Informujemy wolnych klientów, że partia jest już nieaktualna.
Gość	odejście z partii	Usuwamy klienta z list zajętych i umieszczamy na liście wolnych. Wysyłamy gospodarzowi partii komunikat o odejściu gościa.

4.4. Zerwanie połączenia

W przypadku, gdy połączenie z klientem zostanie przerwane serwer musi podjąć pewne działania. Jeśli klient znajdował się w trybie:

- **gospodarza** – powiadamiamy wszystkich gości w partii o jego odejściu i usuwamy partię.
- **gościa** – powiadamiamy gospodarza o odejściu klienta.

Serwer usuwa adres klienta z pamięci.

5. Słownik

Gra - rodzaj zabawy towarzyskiej, u nas internetowej, odbywający się według określonych reguł, na przykład: szachy, chińczyk.

Partia - jest to przebieg pewnej gry, na przykład: partia szachów.

Gospodarz - klient, na którego komputerze odbywa się logika gry.

Gość - klient biorący udział w partii, niebędący gospodarzem.

Logika gry - stan partii, wartości wszystkich obiektów związanych z partią.