Dokumentacja końcowa projektu:

zadanie 1 - gra z interfejsem na przeglądarkę

Jacek Sosnowski, Łukasz Gadawski

1 czerwca 2013

1. **Opis projektu**

2. **Szczegółowy opis gry**
   Po odwołaniu w przeglądarce do adresu serwera gry użytkownik jest proszony o wybór swojego identyfikatora. Domyślnie jest wybrana nazwa „Guest x" gdzie x jest kolejnym numerem – inkrementowanym dla każdego nowego żądania. Możliwa jest zmiana loginu na dowolny, pod warunkiem, że będzie wolny. W przypadku zajętości w polu poniżej pola logowania zostanie wyświetlony komunikat o duplikacie nazwy użytkownika – należy wybrać inną. Po zalogowaniu uwidacznia się widget z podziałem na obszary. Prawy panel zawiera listę użytkowników aktualnie zalogowanych oraz przycisk umożliwiający zaproszenie wybranej osoby do gry. Dolny natomiast informuje na bieżąco o zaistniałych zdarzeniach – logowanie / wylogowanie. Komunikaty (opisujące procedurę rozgrywki lub informujące o błędach) umieszczane są na głównym panelu. Po zaproszeniu do gry ukaże się tam informacja o konieczności oczekiwania na odpowiedź zaproszonego gracza. Ten drugi otrzyma informację o zaproszeniu oraz możliwość akceptacji lub odrzucenia (dwa przyciski). Informacja o decyzji jest przesyłana do osoby zapraszającej i w przypadku akceptacji gra jest rozpoczynana po obu stronach. Rozpoczęcie skutkuje wyświetleniem panelu gry zsynchronizowanego dla obu graczy i umożliwiającego oddanie tylko pojedynczego ruchu w naprzemiennej kolejności, aż do rozstrzygnięcia danej partii. W przypadku zakończenia rozgrywki (np. przez wygraną któregoś z graczy) panel gry zostaje usunięty z jednoczesnym wyprowadzeniem wiadomości o zwycięzcy. Rozgrywka może być również przerwana niespodziewanym wylogowaniem którejś ze stron – druga osoba jest o tym powiadamiana, a sama rozgrywka zostaje przerwana z możliwością przeprowadzenia kolejnej z innym graczem.

3. **Założenia projektowe**
   Aplikacja wykorzystuje przeglądarkę internetową jako interfejs graficzny dla klienta gry. Zastosowana została koncepcja pośrednio zbliżona do cienkiego klienta, ponieważ dane, oraz komunikacja odbywa się poprzez serwer. Zostanie to dokładniej opisane dalej. Serwer gry jest zbudowany na podstawie serwera dostarczonego przez bibliotekę Wt. Biblioteka dostarcza potrzebnej abstrakcji w języku C++ dla wszelkich potrzebnych technologii internetowych, które wykorzystuje aplikacja (JavaScript, Ajax, budowa dokumentu HTML itp.) Natomiast sam wygląd witryny jest modyfikowany z użyciem stylów CSS. Aplikacja zbudowana jest w oparciu o architekturę klient-serwer. Przeglądarka będąca klientem komunikuje się bezpośrednio tylko z serwerem. W przypadku samego procesu serwera w projekcie nie wprowadzono wielowątkowości. Dostęp do zasobów wspólnych jest mimo to synchronizowany bo są one dzielone pomiędzy różnymi sesjami klienckimi. Program źródłowy serwera jest całkowicie przenośny pomiędzy większością platform na jakie są udostępniane wykorzystane biblioteki. Kompilację i uruchomienie pomyślnie ukończono w środowiskach

Windows (wersja 7) oraz Linux (system Ubuntu 13.04). Uruchomienie procesu serwera umożliwia dostęp do gry z dowolnej przeglądarki internetowej, niezależnie od systemu operacyjnego. Pomyślnie działały najnowsze dystrybucje przeglądarek: Firefox, Chrome, Opera. Dla maszyny, która udostępnia serwer gry zakładane jest też posiadanie interfejsu sieciowego wykorzystującego protokół IPv4. Uruchomienie jest możliwe zarówno dla adresów sieci lokalnej jak i dla publicznie dostępnych – obie opcje zostały sprawdzone.

4. **Architektura rozwiązania**
   System składa się z jednego procesu serwera gry, który wykorzystuje serwer Wt::WServer i mechanizm punktów dostępowych (entry points) dostarczone z biblioteki WebToolkit. Oznacza to brak konieczności powoływania w jawny sposób różnych wątków dla każdej sesji użytkownika oraz późniejszego nimi zarządzania. Każdej nowej sesji (wywołaniu adresu serwera z przeglądarki – niekoniecznie jest wymagane bezpośrednie otwarcie strony) jest przypisywany nowy obiekt zarządzający tą sesją i prezentujący widoki dla przeglądarki. W tym przypadku jest to obiekt klasy GameApplication, którego tworzenie jest zarejestrowane dla jedynego w systemie punktu dostępowego. Każda nowa sesja przechodzi przez wszystkie zadeklarowane punkty dostępowe jednocześnie otrzymując wymagany obiekt oraz referencję do obiektu serwera. Biblioteka Wt w sposób niejawny ukrywa wielowątkowość. Nietypowym rozwiązaniem jest stworzenie pustej klasy (Client) do identyfikacji klientów. Każdy obiekt zarządzający grą po stronie klienckiej (GameWidget) dziedziczy po Client. Z kolei wszystkie obiekty powstają w przestrzeni adresowej serwera dlatego wskazanie na Client można traktować jako unikatowy identyfikator klienta. Dzięki dziedziczeniu klasa GameWidget może przedstawiać się serwerowi w łatwy sposób. Oczywiście użytkowników identyfikują nazwy (login) ale mogą one podlegać zmianom w przeciwieństwie do widget'u gry, który ginie dopiero po wylogowaniu. Po stronie klienta (w przeglądarce) w zależności od zaistniałej sytuacji tworzone są potrzebne widget'y. Komunikacja pomiędzy nimi wykorzystuje wzorzec sygnałów i slotów, udostępniany w bibliotece WebToolkit, która z kolei bezpośrednio korzysta z sygnałów biblioteki boost. Nie wszystkie zaistniałe zdarzenia są propagowane do serwera. Jeśli jest konieczność wypisania komunikatu tylko po danej stronie, to oszczędzane jest pasmo i unikana nadmierna komunikacja. Mechanizm sygnałów i slotów w wykorzystywanej wersji stosuje koncepcję pętli zdarzeń znaną również z innych bibliotek (chociażby z Qt). Dlatego wszystkie zdarzenia są kolejkowane i wykonywane w miarę możliwości. Pętle zdarzeń działają po stronie każdego klienta, w przeglądarce dzięki użyciu JavaScript oraz w procesie serwera.

5. **Komunikacja**
   Jak już wspomniano, większość zdarzeń zaistniałych w systemie jest propagowana do innych sesji klienckich poprzez proces serwera z wykorzystaniem mechanizmów Wt. Komunikacja wykorzystuje protokół http (meto-

dy GET i POST) oraz zazwyczaj przeprowadza kompresję. Jej dostępność zależy od konfiguracji środowiska uruchomieniowego – dostępności biblioteki zlib. Ponadto w przypadku dostępności pakietu OpenSSL jest możliwość zapewnienia bezpieczeństwa. Za samą komunikację odpowiada mechanizm wykonywania metod po stronie sesji klienta (post method). Oznacza to że serwer może zlecić wykonanie pewnych operacji obiektom sesji klienckiej nie czekając na ich zakończenie. Wykonuje wtedy wstawienie zadania do pętli zdarzeń klienta (metoda post). W przypadku tego systemu tymi zadaniami była obsługa zdarzeń (events). Za tą obsługę odpowiada metoda GameWidget::processGEvent(). Dlatego podczas inicjowania połączenia klienta z serwerem musi on podać wskazanie na metodę ze swojego obiektu, która będzie przejmować przetwarzanie zdarzeń. Serwer dysponując jednoznaczną identyfikacją sesji klienckich za pomocą wskazań Client* oraz znając metody do których ma przekazywać zdarzenia jest w stanie pośredniczyć w komunikacji. Na pierwszy rzut oka wygląda ona jak zwykłe wywołania funkcji, ale w rzeczywistości te wywołania odbywają się w innych kontekstach i w ogólności na innych maszynach. Z punktu widzenia sieci pomiędzy serwerem i klientem nawiązywane jest połączenie TCP. Dlatego najbardziej kosztowny jest otwieranie strony – Three-way handshake z dodatkiem narzutu czasowego metod na poziomie logiki aplikacji do inicjowania połączenia: tworzenie obiektów sesji, wywołania metod logowania. Połączenie jest utrzymywane dzięki opcji keep-alive. W równych odstępach czasu wysyłane są pakiety heartbeat w celu sprawdzenia istnienia sesji. Koniec identyfikacji sesji klienta poprzez serwer następuje po wylogowaniu (użycie przycisku log out).

6. **Testowanie**
   Wstępny plik testowy stworzony.