05.06.2015r.

Paweł Kamiński
Carlos Zaldivar Batista
Jan Muczyński

Prowadzący: dr inż. Robert Nowak

Liczba poświęconych godzin na projekt: 85

Projekt ZPR
Dokumentacja Końcowa

Temat projektu

Przeglądarkowa gra dwuosobowa kółko i krzyżyk, obsługująca użytkowników na jednym serwerze.

Gra

Ogólna idea

Gra będzie się toczyć na planszy o rozmiarze 16 na 16 pól. Gracze w wybranych przez siebie miejscach wstawiają kółko i krzyżyk. Warunkiem zwycięstwa jest ustawienie 5 takich samych symboli sąsiadujących ze sobą w jednej linii (pionowo, poziomo bądź na ukos). W przypadku, gdy zostaną wykorzystane wszystkie pola oraz żaden z graczy nie spełni warunku zwycięstwa to gra kończy się remisem.

Użytkownicy toczą rozgrywkę, każdy ruch przed przesłaniem do serwera jest sprawdzany pod kątem poprawności. Wykonanie ruchu przez jednego z graczy blokuje możliwość wykonywania ruchu przez drugiego.

Pierwszy ze zgłoszonych graczy ma przydzielane kółko a drugi krzyżyk. Ponadto przy nicku gracza znajduje się licznik jego zwycięstw w danej grze. Po zakończeniu partii istnieje możliwość rewanżu po obopólnej zgodzie.

Grę wygrywa gracz z większą liczbą wygranych partii. Informacje o bieżącej rozgrywce oraz o liczbie wygranych/przegranych partii są przechowywane w pamięci po stronie serwera.

W jednej chwili na serwerze może być rozgrywanych wiele partii pomiędzy wieloma graczami.

Funkcjonalność

Uzyskana funkcjonalność

1) Podstawowa mechanika rozgrywki, plansza, dwóch graczy, liczniki punktów.
2) Uruchamianie oddzielnej rozgrywki dla każdych dwóch kolejnych graczy.
3) Uruchamianie ponowne rozgrywki pomiędzy dwoma graczami(rewanż).
4) Informowanie, w jakiej fazie rozgrywki znajduje się dany gracz(czyja tura, ilość wygranych gier, stan gry).
5) Zaznaczanie na inny kolor pól, które spowodowały zwycięstwo jednego z graczy.
6) Informowanie przeciwnika o tym, że gracz opuścił grę(nie zawsze działa poprawnie). Możliwość szukania nowego gracza do rozgrywki, po naciśnięciu przycisku(odświeża stronę).

Funkcjonalność, na którą zabrakło czasu

1) Uzyskiwanie punktu przez przeciwnika w wypadku zbyt długiego czasu wykonywania ruchu.
2) Zmiana rozmiaru planszy.
3) Chat pomiędzy graczami.
4) Wyszukiwanie nowego gracza w tej samej sesji przeglądarki(przycisk pojawiający się po opuszczeniu rozgrywki przez drugiego gracza odświeża stronę).

Technologie, które zostały użyte

Klient

Klient używa jedynie przeglądarki www w celu zagrania w grę. Interfejs graficzny został napisany w JavaScript(jQuery) połączonym z *HTML5* i *CSS*.

Serwer

Do utworzenia, kompatybilnego z systemami operacyjnymi Linux i Windows, serwera zostało użyte *web2py*, zaś *boost::python* + C++11 zostały zastosowane do stworzenia logiki aplikacji.

System kontroli wersji

Podczas tworzenia aplikacji zostanie użyty system kontroli wersji *git* w serwisie github pod adresem https://github.com/pawelkami/ZPR.

Rozgrywka

Komunikacja z serwerem

Przed rozpoczęciem gry, gracz podaje swój nick. Następnie wywoływana jest poprzez jQuery.ajax() funkcja z Pythona rejestrująca gracza. Po rejestracji gracz czeka do momentu pojawienia się przeciwnika. Gdy to nastąpi, rozpoczyna się gra.

Ruch w grze jest turowy. Gracz, którego jest w danym momencie kolej, stawia swój znak na wybranym przez siebie polu, który jest sprawdzany przez JavaScript po stronie klienta pod względem poprawności(czy nie został postawiony na zajętym miejscu). Następnie ruch jest przesyłany za pomocą jQuery.ajax() do funkcji w Pythonie, która wyszukuje daną grę z listy aktualnie rozgrywanych gier, korzystając przy tym z przydzielonego na początku id gracza. Następnie uaktualnia planszę gry w obiekcie Game zaimplementowanym w C++. Potem wywoływana jest metoda obiektu gry sprawdzająca warunek zwycięstwa i zwracająca aktualny stan gry. Następnie informacje na temat ostatnio postawionego ruchu, liczby punktów oraz stanu gry są przekazywane do obu graczy. W razie wygranej bądź remisu, zwycięskie pola zaznaczane są na czerwono oraz na ekranie obu klientów, pojawia się okienko mówiące o rezultacie gry z przyciskiem umożliwiającym rozegranie kolejnej partii.

Uruchamianie i kompilacja

Linux

Szczegółowa instrukcja znajduje się w pliku README.md. Do kompilacji wymagane są biblioteki boost oraz program scons. Do uruchomienia serwera należy zainstalować web2py oraz Pythona. Katalog z repozytorium należy umieścić pod ścieżką „katalog/do/web2py/applications/game”.

Windows

Do kompilacji jest potrzebny program scons(należy pamiętać również o dodaniu go do zmiennej PATH) oraz biblioteki boost 1.57. Do uruchomienia serwera niezbędny jest web2py oraz Python. Katalog z repozytorium należy umieścić pod ścieżką „katalog/do/web2py/applications/game”. Następnie należy otworzyć linię poleceń w katalogu game i wpisać

$ scons

aby skompilować źródła.

Aby uruchomić serwer, należy przejść do głównego katalogu web2py i uruchomić plik web2py.exe. Następnie wpisać dowolne hasło. Gra powinna pojawić się pod adresem http://127.0.0.1:8000/game.

Testowanie

Liczba testów: 53

Pokrycie kodu testami:

Game.cpp	97.95%
Project_declarations.hpp	100%
Move.hpp	100%
GameList.cpp	100%
Player.hpp	100%
Game.hpp	100%

Testy zostały przeprowadzone przy użyciu *boost::unit_test*.

Do sprawdzenia jakości testów(pokrycie kodu) został użyty program *gcov*.

Procentowy udział języków

Język	Procentowy udział	Liczba linii kodu
C++	78.6%	2454(w tym 1178 testy)
JavaScript	7.8%	323
Python	5.9%	101
CSS	5.3%	284
HTML	1.8%	54

Procentowy udział jest wyliczony według statystyk Githuba. Zawierają się w nim również dodatkowe pliki, potrzebne do projektu, nie napisane przez nasz zespół. Liczba linii kodu odpowiada rzeczywistej liczbie linijek kodu przez nas napisanej.

Napotkane problemy

Największym problemem w początkowej fazie projektu, było znalezienie sposobu na efektywne połączenie dwóch graczy do tej samej gry oraz sposób współdzielenia pamięci między nimi. Zdecydowaliśmy się wtedy na framework web2py, ponieważ spełniał wszystkie nasze oczekiwania i posiadał bogatą dokumentację oraz wiele przykładów wykorzystania w Internecie. Kolejnym napotkanym problemem była synchronizacja, wykorzystanie boost::shared_mutex powodowało problemy z ładowaniem bibliotek do Pythona, więc musieliśmy skorzystać z std::shared_timed_mutex z standardu C++14. Taką operację musieliśmy przeprowadzić na systemie Linux. Na Windows skorzystaliśmy ze zwykłych Mutexów.

Screenshoty z gry