Tanks

control tank and fight with your friends!

1. Wstęp

Gra Tanks powstała jako projekt zaliczeniowy z przedmiotu Projektowanie aplikacji internetowych. Mieliśmy za zadanie stworzyć grę sieciową z założeniami przedstawionymi poniżej:

- gra zaklada nieskonczona liczbe graczy lub gdy ze wzgledu na reguly to nie ma sensu, gre w wielu pokojach rownoczesnie
- gra zakłada przynajmniej dwie równoległe formy komunikacji (np. gra właściwa plus chat)
- komunikacja pomiedzy klientami (graczami) odbywa sie zawsze za posrednictwem serwera
- jakaś część projektu musi być napisana w języku Java i bazować na połączeniach TCP/IP (sockets)
- serwer jest odporny na nieprawidlowe polecenia przesylane przez klienta
- konfiguracja serwera w pliku XML.
- nalezy dostarczyc takze zbior testow jednostkowych (junit w najnowszej wersji, dodatkowy plus za selenium - gdyby projekt miał TAKŻE warstwe WWW)
- należy przygotować dokumentację uwzględniającą wszystkie w.w. aspekty.
- lista gier która NA PEWNO nie może zostać wykonana w ramach zaliczenia projektu (nie jest to lista zamknieta - do ustalenia z prowadzacym), oczekuje się projektów bardziej ambitnych: ping-pong, warcaby, szachy, statki, wisielec, wszystkie gry karciane

Zgodnie z wymaganiami projekt jest napisany w architekturze Klient-Server i przyjmuje zasadę nieskończonej liczby graczy na mapie.

2. Jak uruchomić?

Uruchomienie należy zacząć od servera gry – TanksServer.jar. Domyślnie jest on skonfigurowany na porcie 1234, lecz można to zmienić w konfiguracji serwera (config.xml, szczegóły w rozdziale 4). Uruchomiając aplikację nasz serwer jest już gotowy do pracy. Zero skomplikowania!

Jeżeli już uruchomiliśmy serwer, potrzebujemy teraz jakoś się z nim połączyć. Uruchamiając aplikację TanksClient.jar uruchamiamy klienta gry Tanks. Rozpoczynamy w losowym miejscu na mapie z ilością życia na poziomie (domyślnie: 1000.0, do ustawienia w pliku config.xml) punktów. Gracz może jednym strzałem zadać od 20 do 50 punktów obrażeń. Posiadając poniżej 0.0 punktów życia – giniemy. Ginąc – nasze okno aplikacji zamyka się. Podczas kolizji czołgów również ginemy – my oraz przeciwnik. Aby kontynuować grę musimy ją uruchomić ponownie.

3. Klawiszologia

Czołgiem sterujemy przy pomocy czterech klawiszy (A, W, S, D) oraz SPACJĄ.
A – w lewo
W – w górę
S – w dół
D – w prawo
SPACJA – strzał
ENTER – wyślij wiadomość

4. Konfiguracja

Serwer
W pliku config.xml mamy trzy możliwości konfiguracji.
Port (domyslnie: 1234) – port serwera na którym zostanie on postawiony.
DefaultHP (domyślnie: 1000) – domyślna początkowa ilość życia każdego czołgu.
GameSpeed (domyślnie: 5) – prędkość gry (przemieszczania się bomb/czołgów).

Klient
W pliku config.xml mamy trzy możliwości konfiguracji.
Port (domyślnie: 1234) – port pod którym aplikacja kliencka ma szukać serwera.
Hostname (domyślnie: localhost) – hostname pod którym aplikacja ma szukać serwera.
Nickname (domyślnie: null) – nickname osoby klienckiej (gracza). W przypadku wartości null – generuje nazwę z puli i ją ustawia.

5. Opis projektu TanksServer.jar

Main.java
Główna klasa aplikacji serwera, posiada trzy pola oraz kilka metod wspomagających pracę.

Vector<ClientHandler> activeClients – vektor zawierający instancje klasy ClientHandler (opis klasy znajduje się poniżej).
int numberOfClients – liczba wszystkich dotychczasowych graczy (w dalszej części ID gracza; zerowany przy starcie aplikacji).
int port – port serwera (na tym porcie aplikacja serwerowa nasłuchuje chcących uzyskać połączenie klientów).
float defaultHP – przechowuje domyślną startową ilość życia gracza.
float gameSpeed – przechowuje prędkość gry wczytaną z konfiguracji.

Vector<ClientHandler> getActiveClients() - metoda wysyłająca aktualną listę klientów (tych aktualnych oraz już nieżyjących graczy; wszystkich od momentu startu aplikacji serwerowej)
sendToAll(String text) – metoda wysyłająca do każdego AKTYWNEGO gracza wiadomość pobraną z argumentu text
sendData() - metoda wysyła do wszystkich AKTYWNYCH graczy informację na temat pozycji, właściciela, kierunku, życia oraz widoczności wszystkich czołgów
findPlayerByNickname(String name) – metoda aktualnie niewykorzystywana; przeszukuje vektor graczy w poszukiwaniu nazwy gracza podanego w argumencie. Jeżeli znaleziono – zwraca jego identyfikator, jeżeli nie – zwraca liczbę -1
main(String[] args) – metoda główna aplikacji, zawiera kilka obowiązkowych pól oraz pętlę główną, która jest odpowiedzialna za przydział przestrzeni serwerowej po zapytaniu klienta oraz obsługę jego żądań – dodanie nowego gracza, przesłanie wiadomości na chatboxie, przesunięcie czołgu, strzał oraz zadane obrażenia.
void loadXMLFile() - metoda wczytująca plik konfiguracyjny config.xml.

ClientHandler.java
Klasa poboczna przechowujące informacje na temat gracza. Znajdziemy tam pola takie jak:
String nameOfClient – nazwa klienta
DataInputStream dis – uchwyt do odbioru informacji od klienta
DataOutputStream dos – uchwyt do wysyłki informacji do klienta
boolean connected – zmienna przechowująca informację, czy dany gracz jest jeszcze połączony z serwerem
int posX, int posY – pozycja czołgu gracza
int angle – obrót czołgu (0 – góra, 1 – doł, 2 – prawo, 3 - lewo)
float health – ilość punktów życia czołgu gracza
int idTank – identyfikator czołgu/gracza (Z racji, że jedna sesja = jeden czołg, te zmienne identyfikują się nazwajem)

Najważniejsze metody klasy:a
void isCollision() - sprawdza ewentualną kolizję czołg-czołg; jeżeli prawda – zabija graczy obydwóch czołgów.
move(String direction) – na podstawie kodu binarnego (1000 - góra, 0100 – doł, 0010 – prawo 0001 – lewo) zmienia kierunek (obrót) czołgu na polu bitwy.
ClientHandler(String name, int idTank, Socket s, DataInputStream dis, DataOutputStream dos, int x, int y) – konstruktor klasy; tworzy nowe połączenie/czołg

Klasa posiada inicjalizację nowego wątku, który ma za zadanie sprawdzać połączenie z graczem, odpowiadać na jego zapytania i w razie ewentualnego rozłączenia gracza – wykreśla go z listy aktualnych graczy.

Komunikaty wychodzące z serwera:
- NewMessage nazwaGracza (ID: idGracza): Wiadomość
- NewBomb idCzołgu
- NewPos posX|posY|idCzołgu|kierunek|punktyŻycia|czyPołączony
- idCzołgu (przy inicjalizacji połączenia)

Komunikaty przychodzące do serwera:
- NewPlayer nazwaGracza
- NewMessage nazwaGracza (ID: idGracza): Wiadomość
- TankMove kierunek
- Shot
- dmg idGracza|obrażenia

6. Opis projektu TanksClient.jar

Main.java
Główna klasa aplikacji klienta, posiada kilka metod wspomagających pracę.

Jako, że klient musiał być aplikacja okienkową, a ja wybrałem bibliotekę javafx – większość to inicjalizacja okna.

Po uruchomieniu aplikacja domyślnie losuje nazwę użytkownika z puli nazw umieszczonej w kodzie oraz takową ustawia danemu użytkownikowi. (Możliwa jest wersja z podawaniem nazwy użytkownika jako argument wywołania, lecz dopiero po odpowiedniej zmianie w pliku konfiguracyjnym).

Klasa zawiera również kilka nasłuchiwaczy takich jak np. zamknięcie okna powoduje rozłączenie z serwerem oraz obsługę klawiszy do sterowania czołgiem. Każda z tych czynności wysyła odpowiedni komunikat do serwera, który obsługuje czynność po jego stronie.

Controller.java

Jedna z równie ważnych klas. Zawiera wszystkie metody potrzebne do komunikacji klient-serwer oraz przechowuje lokalnie informacje na temat innych graczy (ich pozycji, stanu punktów życia) potrzebnych do wyrenderowania obiektów na mapie gry.

Pola tej klasy:
int port – port pod którym klient ma szukać serwera
String hostname – nazwa serwera (Domyślnie localhost)
boolean running – czy połączenie z serwerem jest zachowane (w przypadku rozłączenia – zmienia się na false)
List<String> chatList – lista przechowująca wiadomości na chatcie
List<Tank> tankList – lista przechowująca instancje czołgów graczy
List<Bomb> bombList – lista przechowująca instancje bomb (strzałów) wysłanych przez graczy
String nickName – nazwa gracza
int idTank – identyfikator gracza/czołgu (Z racji, że jedna sesja = jeden czołg, te zmienne identyfikują się nazwajem)
DataOutputStream dos – uchwyt połączenia z serwerem (klient=>serwer).
DataInputStream dis – uchwyt połączenia z serwerem (serwer=>klient).
float gameSpeed – prędkość gry
float defaultHP – ilość startowej ilości życia

void sendMessage() – metoda odpowiedzialna za wysłanie wiadomości z TextFielda do serwera, który następnie roześle to do reszty graczy i umieści w liście chatList każdego z graczy osobno.
void sendToServer(String mess) – główna metoda komunikacji klient-serwer; wysyła wiadomość podaną jako argument do serwera.
boolean findId(int id) – przeszukuje listę czołów (tankList) w poszukiwaniu podanego jako argument identyfikatora czołgu. Jeżeli znajdzie – zwraca true, jeżeli nie – false.
String generateChat() - metoda odpowiedzialna za generowanie chatu. Ma za zadanie ograniczać liczbę wiadomości (kolejka FIFO o pojemności 30 wiadomości). Zwraca zformatowaną treść chatboxa, prosto do odpowiedniego labela.
boolean checkCollision(int idBomb, int idTank) – wykrywa kolizję bomby (strzału) z czołgiem gracza. Jeżeli taka zaistnieje – zwraca true, jeżeli nie – false.

Klasa zawiera inicjalizację wątku pobocznego, który ma za zadanie obsługiwać całą komunikację – od połączenia gracza z serwerem, poprzez sterowanie czołgiem w trakcie gry, a kończąc na rozłączeniu po jego zniszczeniu.

Tank.java
Klasa przechowuje lokalne informacje o czołgach graczy (potrzebne do późniejszego wyrenderowania mapy gry).

Bomb.java
Klasa przechowuje lokalne informacje o bombach graczy (potrzebne do późniejszego wyrenderowania mapy gry).