Ticket to Ride – Tutorial

Introduction

Ticket to Ride is the implementation of the popular board game “Ticket to Ride”, a cross-country train adventure game. This tutorial describes how to run the server and client software. It is based on Eclipse 3.6(Helios)

Pre-conditions

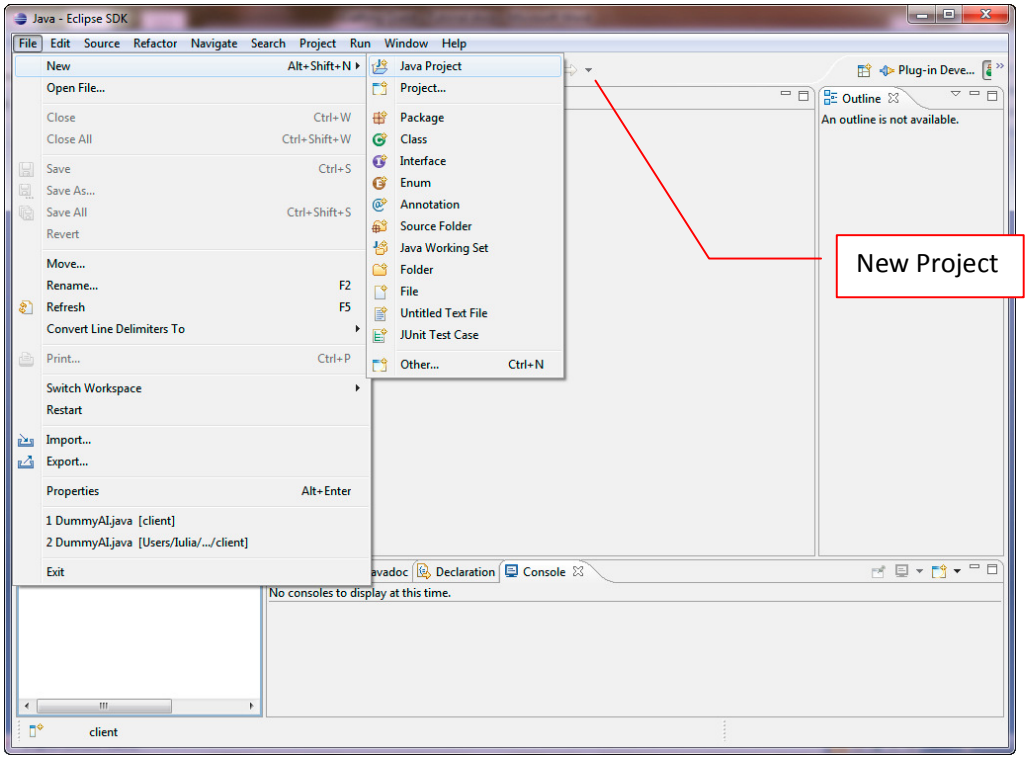
The following assumes that you already know how to use Eclipse IDE.

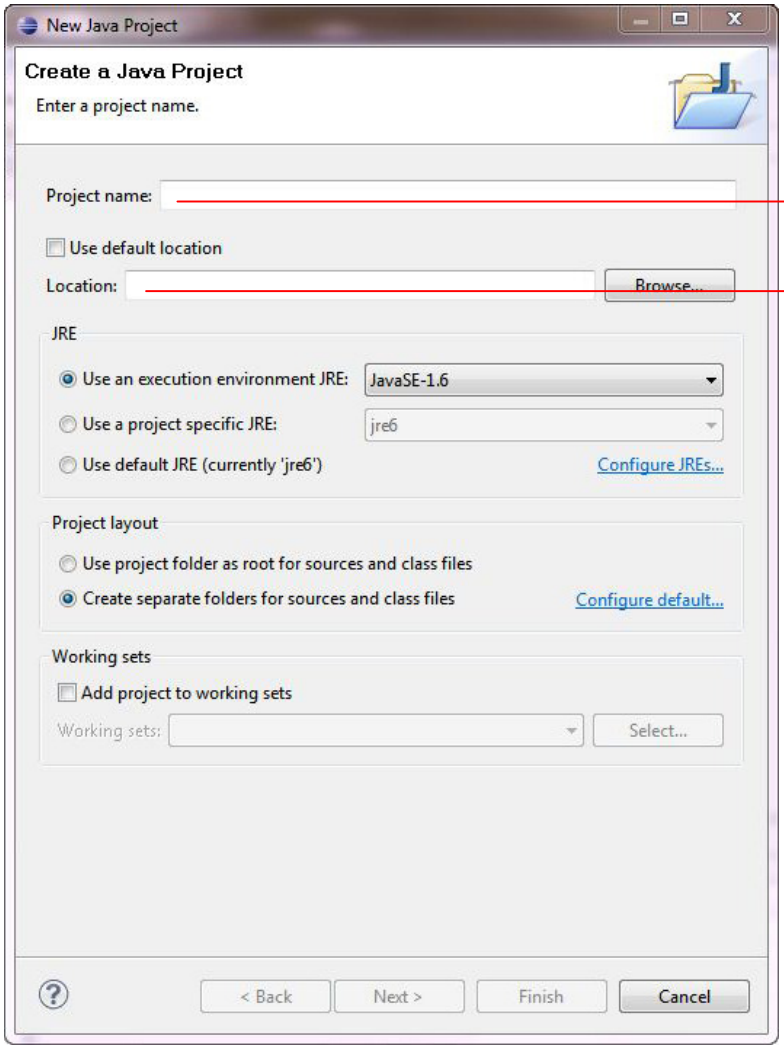
Running the Clients

Java Client

In order to run your solution, you should create a new Java project, setting the folder extracted from the **TicketToRideJavaClient.zip** archive as default location (uncheck *Use default location* and set the *Location* field)

Set your desired name for the project in the *Project Name* field and press *Finish*.





Insert desired name for the project here

Insert folder extracted from the client archive here

Next, create a new instance of your solution in the main class, TicketToRideJavaClient.java, having the following parameters:

- UserName – name of the user
- Pass – password for the user
- AgentName – name of your agent
- ThreadName – name of the thread
- Host – server address
- Port – server port

like in the example:

```
TTRClient t = new RandomSolution ("user", "user", "agent", "", "192.168.1.2", 1337);
```

Your solution must extend the TTRClient class and implement the abstract method myMethod.

C++ Client

Windows

In order to run the C++ Windows version of the client, open the provided Microsoft Visual Studio 2008 project file in MS Visual Studio, build and run. You will need to copy 'pthreadVC2.dll' to either your system folder or your debug folder in order for the application to work.

Linux

In order to run the C++ Linux version of the client, you need to compile and link all source files:

g++ -Wall -g -c *.cpp

g++ *.o -o client –lpthread

Then run the application: ./client

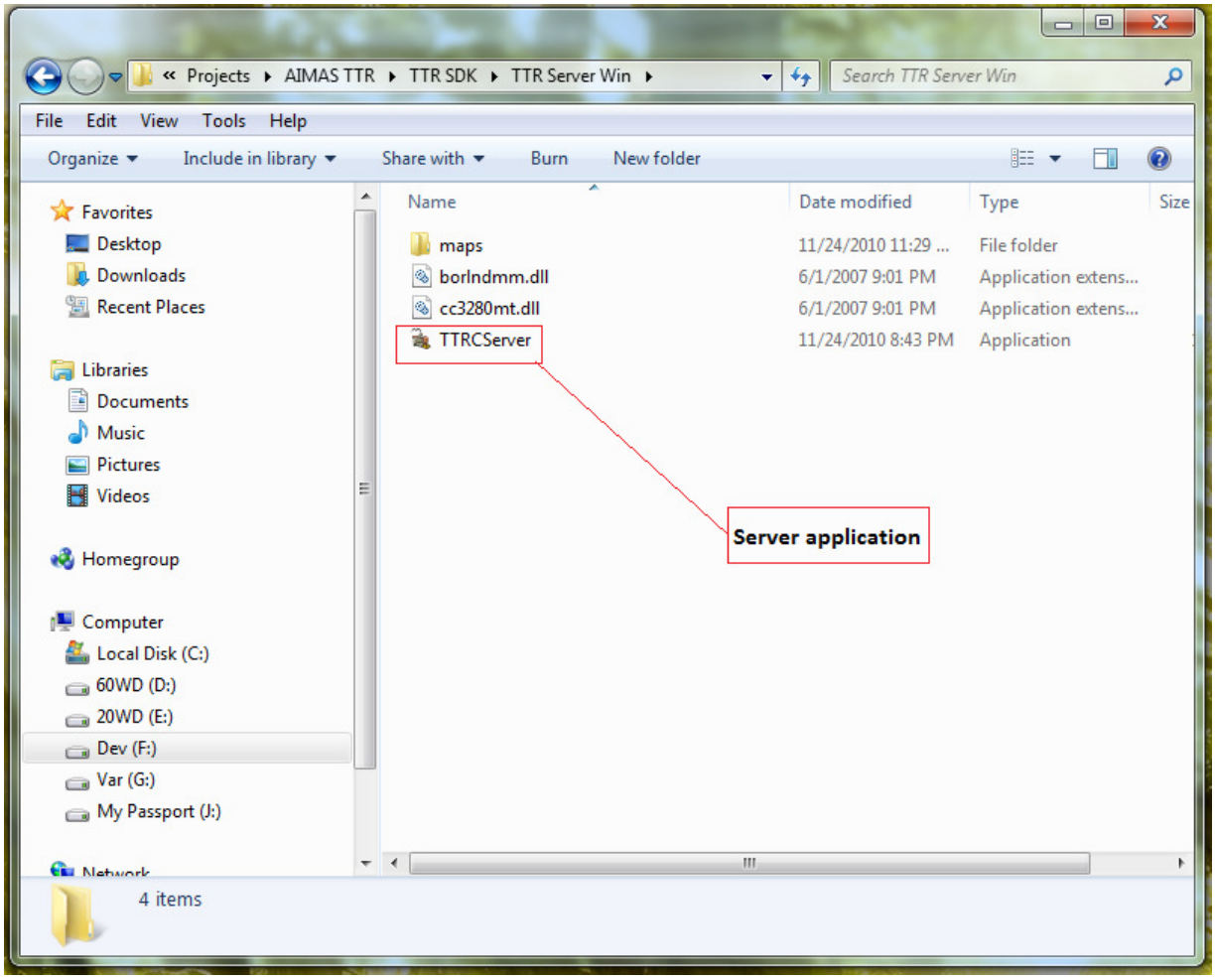
Also a makefile is provided to do this actions: run make.

Note: pthread library comes with glibc

Once started, the client application will run its main procedure (int main(int argc, char **argv) for C++ and void main(String args[]) for Java), which by default runs RandomSolution (TM). The RandomSolution connects to the server and joins the game. RandomSolution will play by attempting, in each turn, to perform a random action until it succeeds.

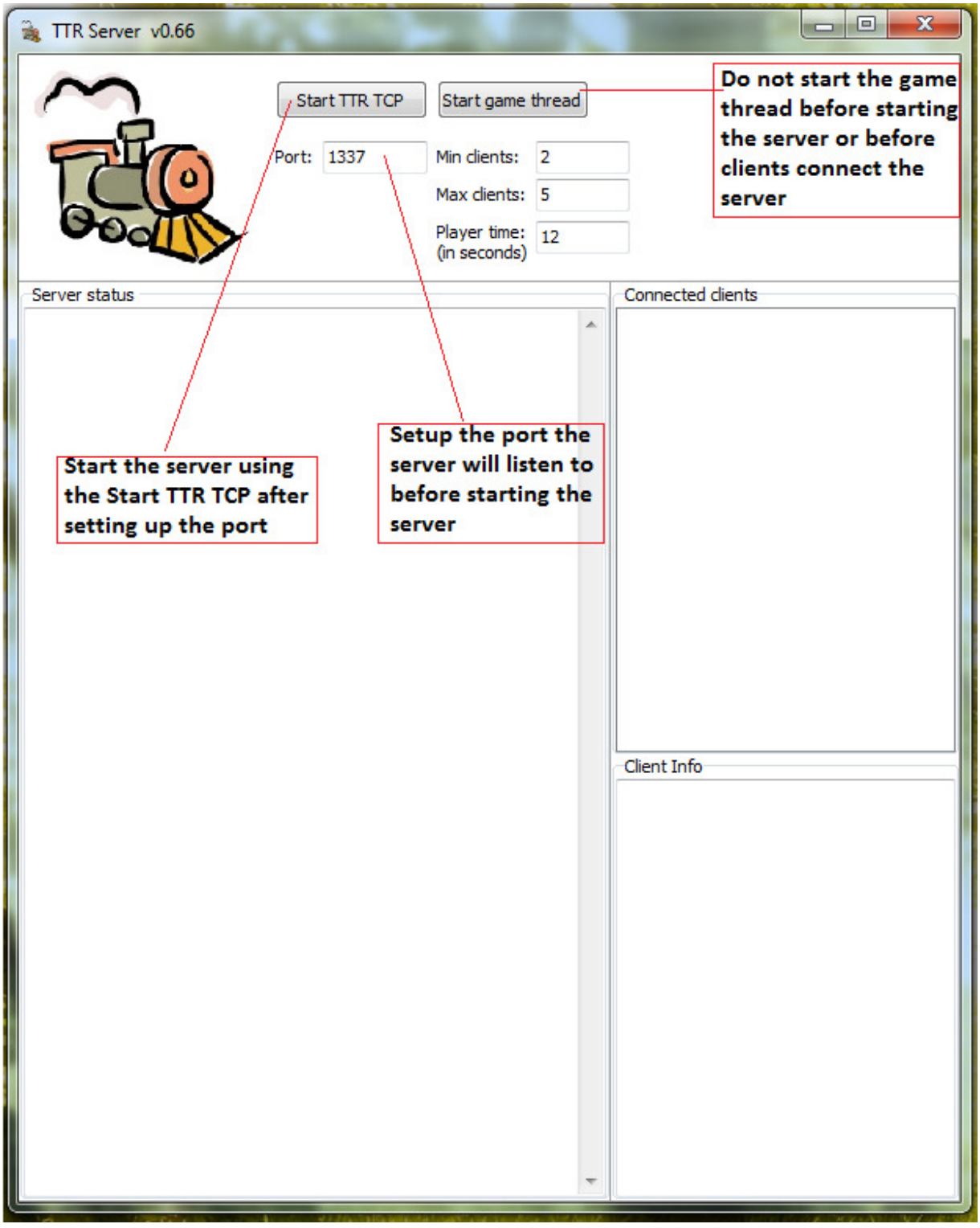
Setting Up and Running the Server On Windows

Copy the Server application file together with the two accompanying .dll files and the map folder to the same location.

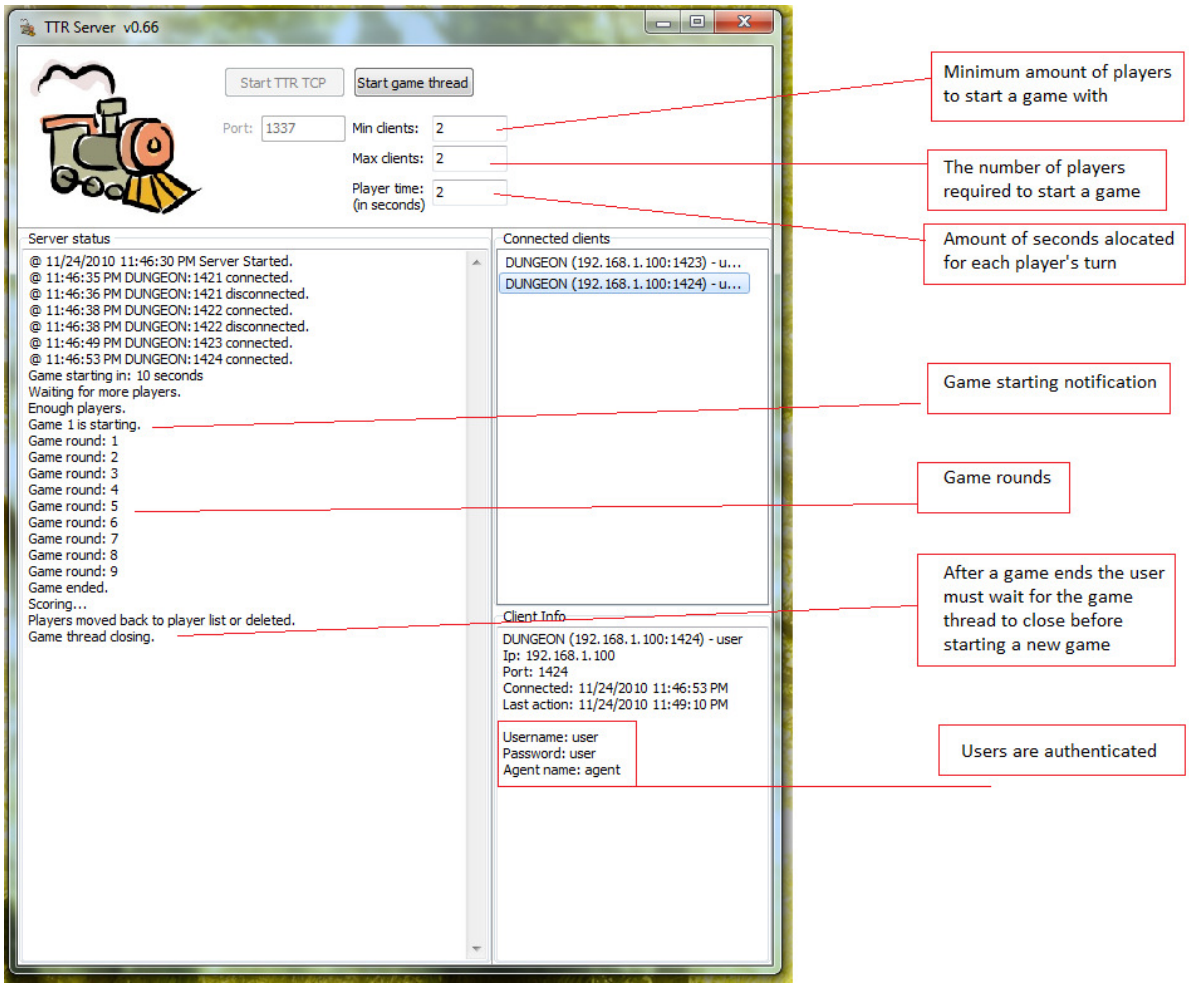


Run the application by executing TTRCServer.exe

In order to start the server you need to choose a port and then hit the *Start TTR TCP* button. The default port is 1337.

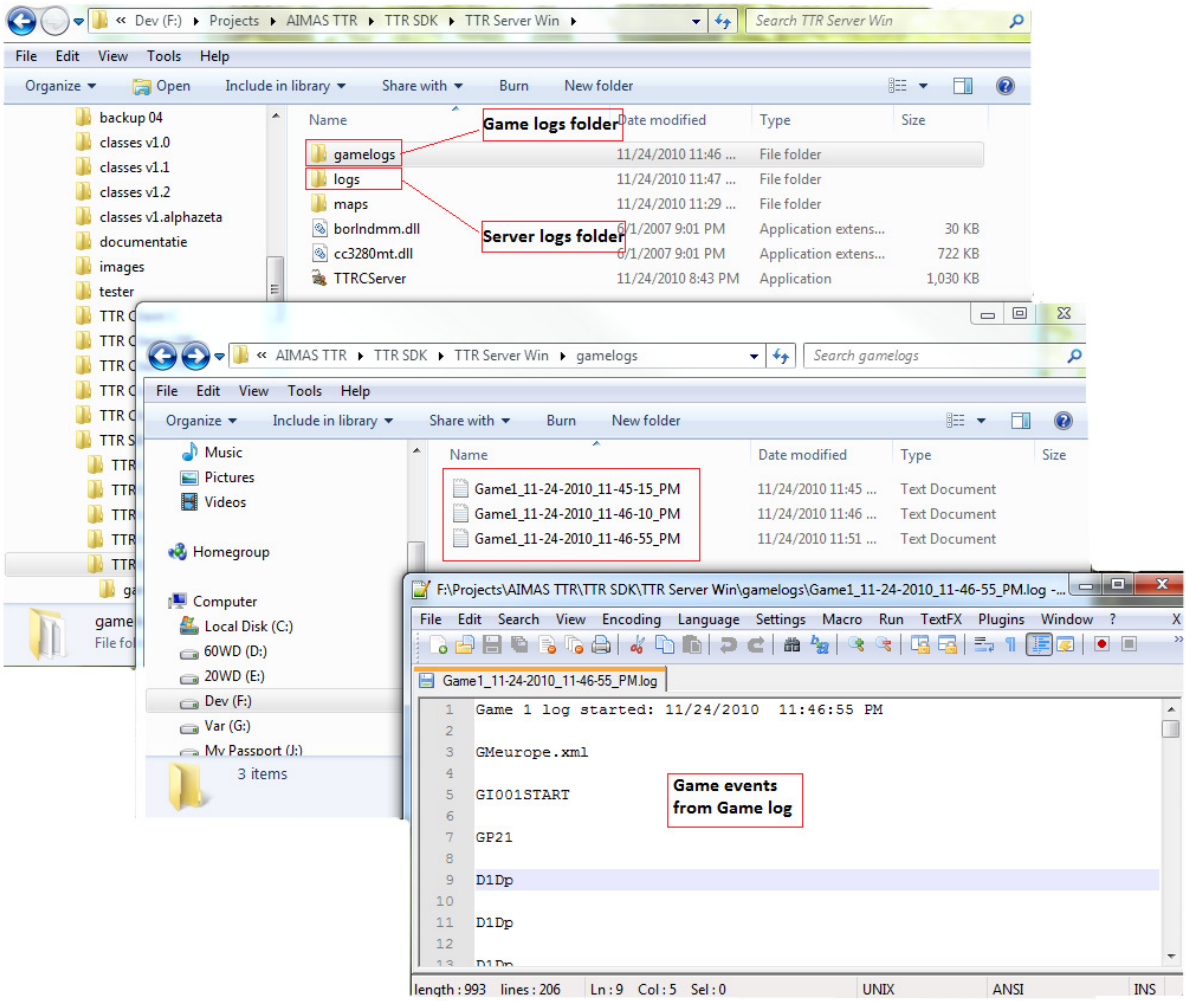


Once the server started and clients connect to it, set up the Minimum and the Maximum Number of Players for a game. Select the amount of time for each player to have and *hit Start game thread*.



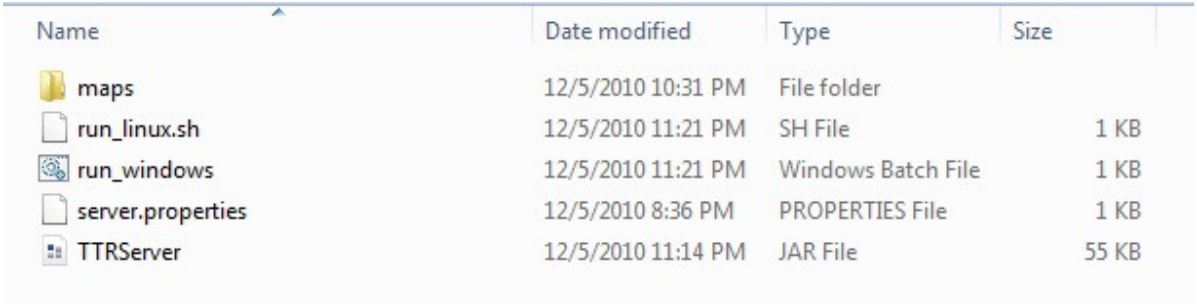
This is how the status of a game would look like after 1 game.

In order to see the results of a game you can open the GAME LOGS folder that will appear in your SERVER APPLICATION Folder, after the first game.
For server logs regarding connections, commands sent and possible errors or exceptions you should check the LOGS Folder.



Setting Up and Running the Server On JAVA

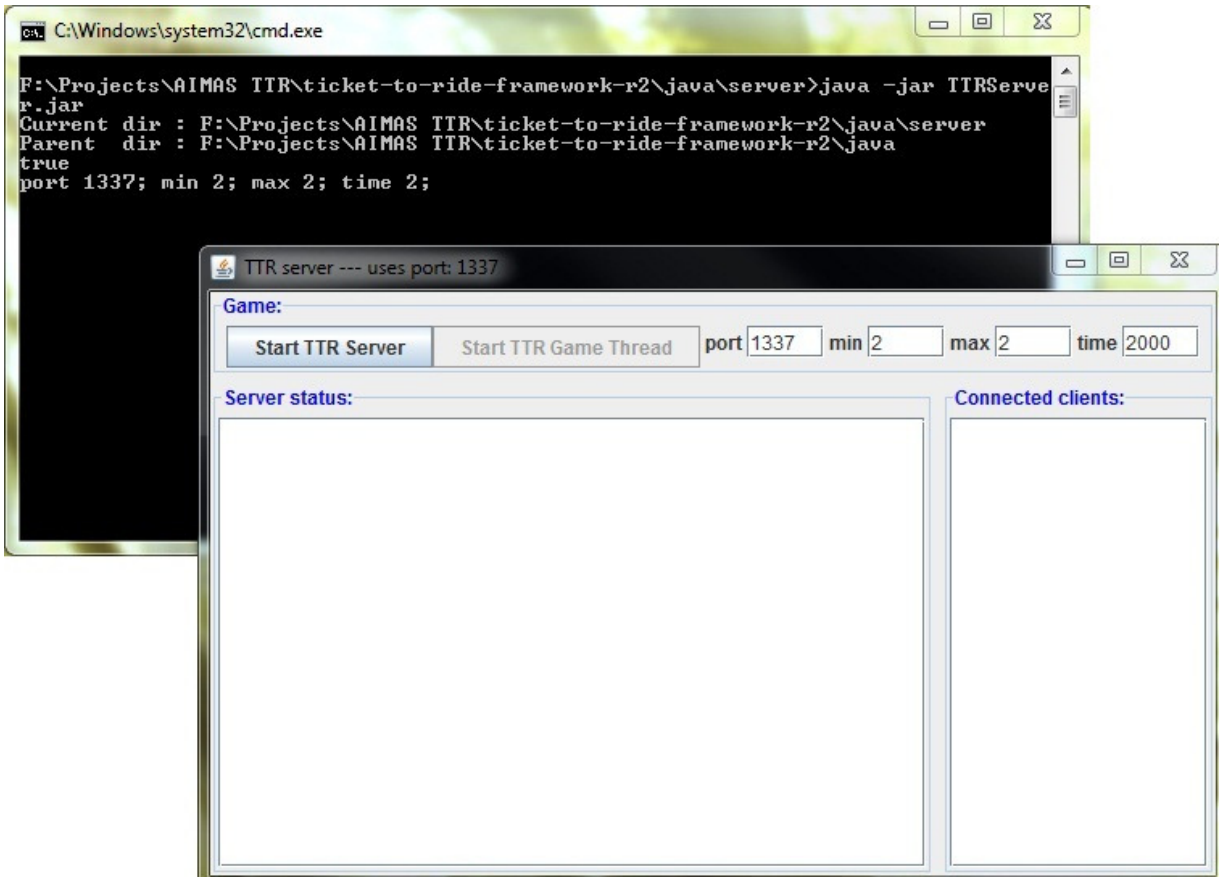
Copy the Server JAR application file together with the server .properties, maps folder, and the appropriate batch file, depending on your system – run_windows.bat (for Windows) or run_linux.sh (for Linux).



Name	Date modified	Type	Size
maps	12/5/2010 10:31 PM	File folder	
run_linux.sh	12/5/2010 11:21 PM	SH File	1 KB
run_windows	12/5/2010 11:21 PM	Windows Batch File	1 KB
server.properties	12/5/2010 8:36 PM	PROPERTIES File	1 KB
TTRServer	12/5/2010 11:14 PM	JAR File	55 KB

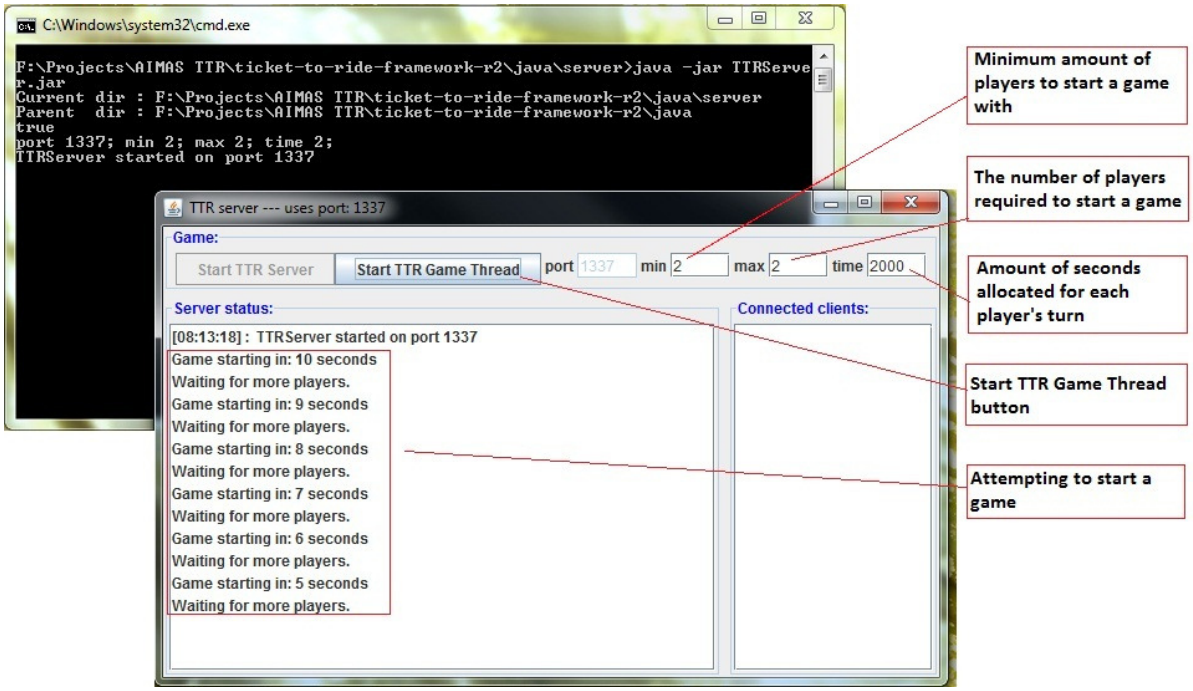
Run the server application by executing the batch file corresponding to you system.

In order to start the server you need to choose a port for the server to listen to, and then click the *Start TTR Server* button.



Note: If the console that started your application is closed, it will shut down the application.

You will be unable to start a game if the server is not started and waiting for connections. Once the server started and clients connect to it, set up the Minimum and the Maximum Number of Players for a game. Select the amount of time for each player to have and hit *Start TTR game thread*.



In order to see the results of a game, you can open the GAME LOGS folder that will appear in your SERVER APPLICATION Folder, after the first game.
For server logs regarding connections, commands that have been sent and possible errors or exceptions, the LOGS Folder should be checked.

***Note*:** Once a game started you will have to wait for it to finish in order to start another. A new game can be started once the *"Game thread closing."* message is printed. Clicking the *Start TTR Game Thread* button will have no effect until the running game thread closes.

Developing a solution

To develop a solution / program of your own, you will be required to use our protocol which will be available for download on the website.

In order to develop a solution for Ticket to Ride you have to extend the TTRClient class (TTRClient.java or TTRClient.cpp) and implement the abstract/pure virtual method myMethod(). This method will be invoked every time it is your turn and should contain the solution's logic

Note: the very first thing you need to do is reject at least one mission from your starting missions.

You must call the **TTRClient(String userName, String passWord, String agentName, String hostName, int portNumber)** super constructor in your constructor.

The complete documentation for data structures is available in the doc subfolder of your chosen project (open index.html), as well as a documented and commented sample solution (RandomSolution.cpp/.java).

Run your solution inside the main procedure. Edit **public static void main(String[] args)** in TicketToRideJavaClient.java or **int main(int argc, char **argv)** in main.cpp.

We would also like to suggest double checking that it is your turn before attempting to perform any actions (the boolean **myTurn** variable should be true).

We also recommend waiting 100 milliseconds between calling the start and go procedures.

JAVA using the TTR JAVA API

TicketToRideJavaClient.java

```
package source;

public class TicketToRideJavaClient {

    public static void main(String[] args) {

        TTRClient t = new RandomSolution(
            "JavaUser", "JavaUser", "JavaAgent", "127.0.0.1", 1337);
        t.start();

        try {Thread.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();}
        t.go();
    }

}
```

C++ (Windows) using the TTR C++ API

main.cpp (Windows)

```
#include <conio.h>
#include "RandomSolution.h"

int main(int argc, char **argv)
{
    if (Socket::initSocketLib()!=0)
    {
        printf("Error initialising socket library\n");
    }

    RandomSolution *s = new RandomSolution(
        "CppUser", "CppUser", "CppAgent", "127.0.0.1", 1337);

    Thread::sleep(100); s->start();

    Thread::sleep(100); s->go();

    _getch(); delete s; s->stop();

    Socket::cleanupSocketLib();

    return 0;
}
```

Note: we use _getch() so that main will not return and kill our threads.

C++ (Linux) using the TTR C++ API

main.cpp (Linux)

```
#include <stdio.h>
#include "RandomSolution.h"

int main(int argc, char **argv)
{
    RandomSolution *s = new RandomSolution(
        "LinuxUser", "LinuxUser", "LinuxAgent", "192.168.65.2", 1337);

    Thread::sleep(100); s->start();

    Thread::sleep(100); s->go();

    char tmp; scanf("%c", &tmp);
    delete s; s->stop();
    return 0;
}
```

Note: we use scanf() so that main will not return and kill our threads.

Submitting Solutions

The solution can be submitted by filling in the submission form available on the website. Teams must submit their solutions in following form:

- the client solution will be sent as an executable file (.exe or .jar)
- any other external libraries needed by the solution in order to run correctly (for example a .dll file) will be sent together with the binary file in the same folder.

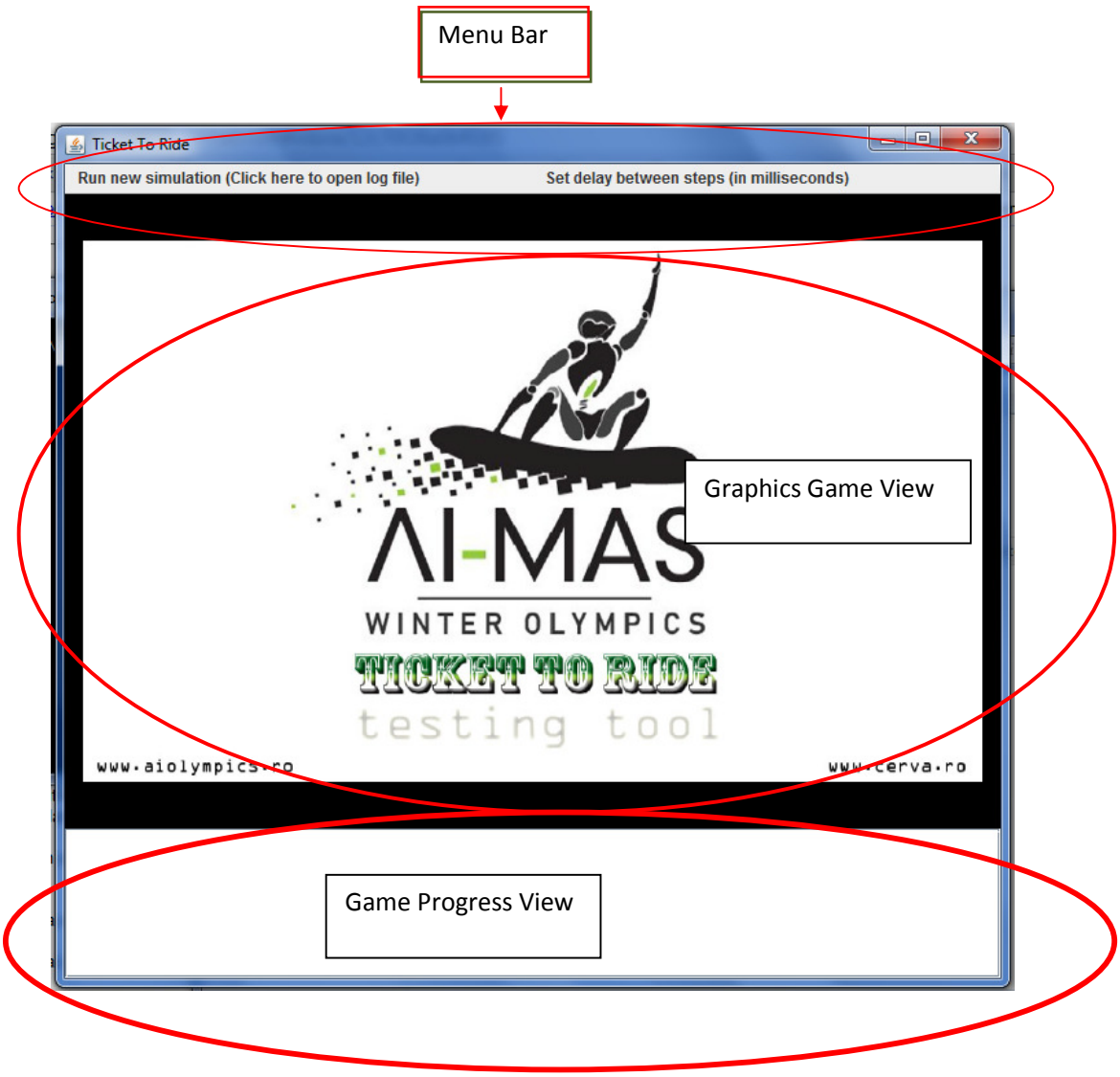
Running the Graphical User Interface (GUI)

A graphical user interface is provided in order to make it easier to view log files.

The graphical user interface comes with two scripts, one for windows, and one for linux. Run the script, click Run new simulation and navigate to the log file you wish to visualize. You may also set the delay between steps.

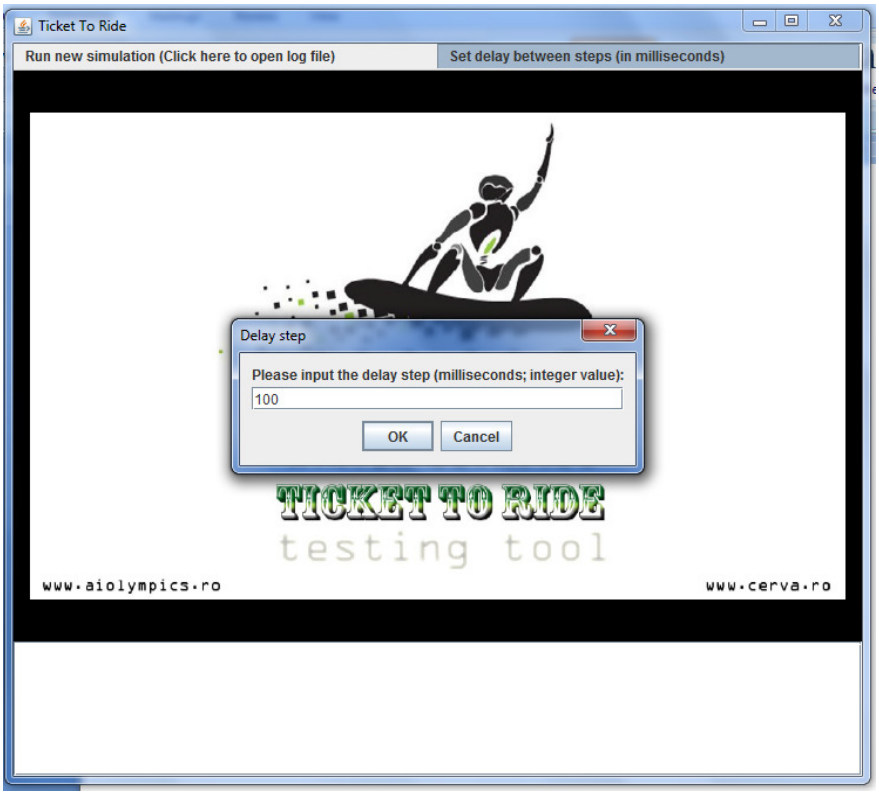
In order to run the GUI application, Java3D must be installed on the computer. You can download the appropriate version here: https://java3d.dev.java.net/binary-builds.html.
After installing Java3D you can run the GUI application by double-clicking on the appropriate executable file, depending on the OS (Windows or Linux).

The following window will pop up:

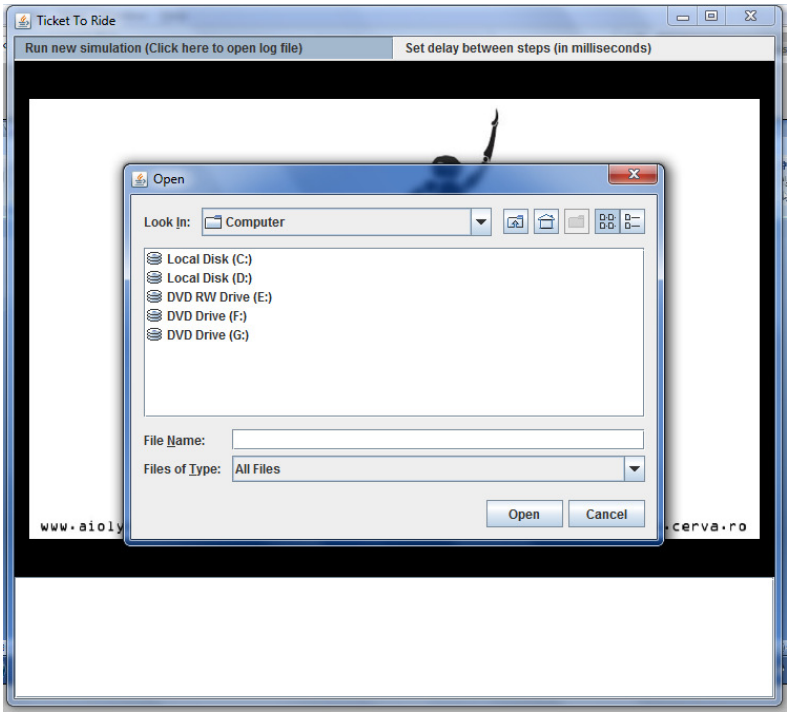


In the Graphics Game View, you can follow the changes on the map in real time.
In the Game Progress View, you can observe what actions are being performed by each player, their current state and how much time they have left.
The menu bar has 2 options:

- Set delay steps: allows you to control the delay with which the actions performed will be read from the log file and displayed in the bottom part of the application window.



- Run new simulation: opens an explorer window, allowing you to select the log file for the match.



The game simulation starts after a log file is selected.