

2009

# Virtual War Games Manual

Version 1.0

In this Manual the description of introducing War games environment is presented. The server, communication, sensor data and even running a sample code is explained completely in this document.

Sanaz Taleghani, Maziar Ahmad Sharbafi, Edriss Esmaeeli, Omid  
Amirghiasvand

[Type the company name]

8/30/2009



## Table of Contents

1. Introduction .....	3
1.1 Background .....	4
2. System prerequisites and run server .....	4
2.1 Simple Client.....	7
3. Communication (Message and Command) .....	7
4. Sensor Simulation.....	11
5. Configuration files.....	12



## 1.1 Background

Virtual War Game is a competition that is part of the Simulation league to provide an opportunity for creating a virtual game environment to play a game with some critical purposes. Its main purpose is to explore and to map by robot utilizing some sensor. The new challenge that may be cause difference between this league and similar league such virtual robot is how robot can prevent the hit of shoot of opponent's robot and how robot can be enhanced the motion reliability at the least risk. Each Team is trying to Conquest opponent team by using better algorithms.

This game had prepared an environment allow testing of the artificial intelligence algorithms and improving them.

## 2. System prerequisites and run server

Virtual War Game (VWG) server requires some software components to operate. Please download and install the following software components on your computer.

Operating System (Windows XP/Vista):

- Microsoft.Net.Framework.3.5, which you can download at (<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=333325fd-ae52-4e35-b531-508d977d32a6>)
- Microsoft.VisualStudio.9.0.x86 redistributed package, Just go into below site for download it (<http://www.microsoft.com/downloads/details.aspx?familyid=32BC1BEE-A3F9-4C13-9C99-220B62A191EE&displaylang=de>)
- DirectX-March09-Redist (<http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=0cf368e5-5ce1-4032-a207-c693d210f616>)  
or  
DirectX\_Nov2007\_Redist, (<http://www.microsoft.com/downloads/details.aspx?FamilyId=1A2393C0-1B2F-428E-BD79-02DF977D17B8&displaylang=en>)

**Note:** If you encounter any problem about missing d3dx9\_?.dll , you must download and copy d3dx9\_?.dll in the directory of System32 to run server completely.

The server and clients can be run on different machines and all the clients exchange data with the War Game Server through the network. Players would use client programs to connect to the server during the game. Once all players/clients have connected to server and send the INIT command, the game is started and their robots are created on simulator. The program client controls the robots on the simulator.

Please do the following run instructions:

1. Install all the prerequisites on your system with the considering component, if your system does not equipped them.
2. At first run the server.exe with the default configuration, if you want to change it please see the configuration section.
3. Choose the rendering sub system between Direct3D or OpenGL. It is recommended to choose the Direct3D9 rendering subsystem.
4. If you want to change the rendering system option, click on each option you want then choose the favorite status.
5. Click on Ok.
6. After run server, the clients can connect to it, and receive sensor data.

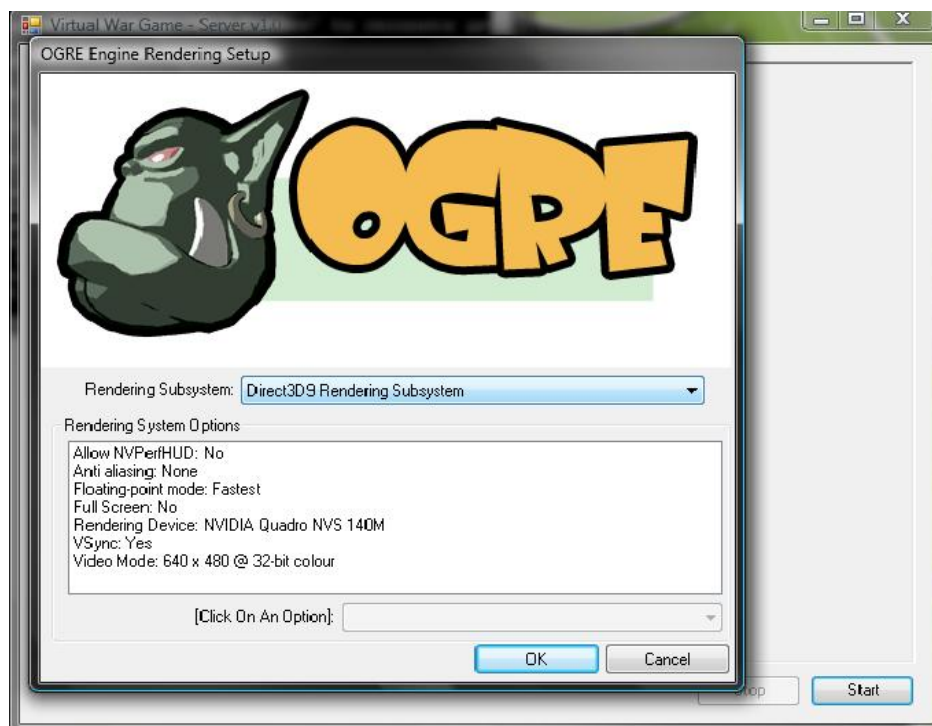


Figure 2: optional controls

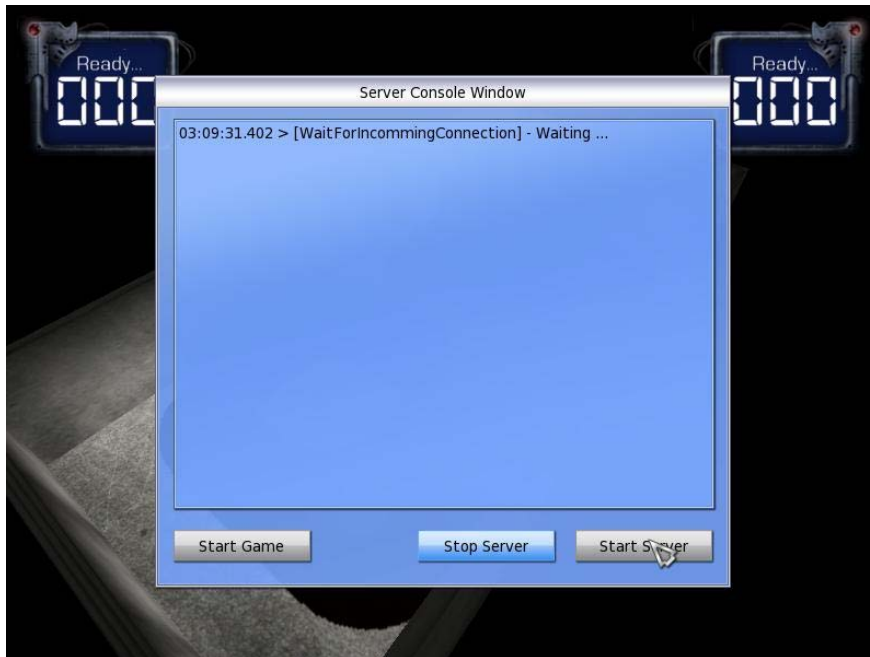


Figure 3: Run the server v1.0

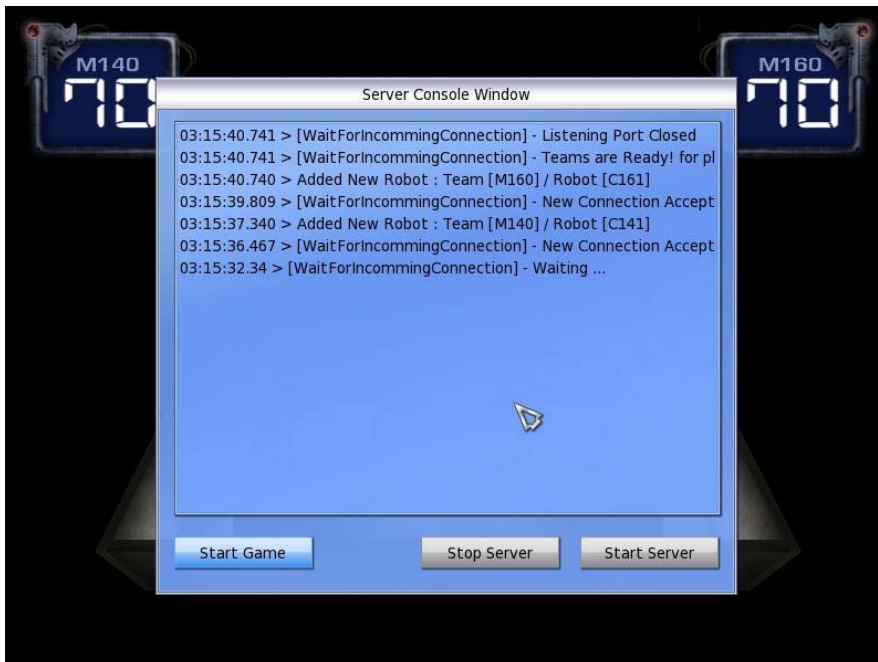


Figure 4: After all team connected successfully

After the server is started, by clicking on **Start Game** button you can change the viewpoint to any place in the simulator. If you inactive the simulation window, rendering will be stopped. If you want to

switch back to simulation window, you can use Alt + Tab keys. There are four short key to change the view. Push the keys “W” ,”A”,”S ”,”D ” to go up ,left, down, right respectively.

## 2.1 Simple Client

Here we prepare a sample of client that you can use it to send any commands to the server, and it will display all the messages that come from the server to you. At first you have to run server, please see mentioned instructions for information on starting server.

The client is executed with the following steps:

1. Install the Microsoft.Net.Client.3.5 which you can download at <http://www.microsoft.com/downloads/details.aspx?FamilyId=8CEA6CD1-15BC-4664-B27D-8CEBA808B28B&displaylang=en>
2. Run the client.exe.
3. Click "Connect" button to connect to the server
4. Send INIT command to create robot on simulator, it is possible that you write your favorite team name and robot name. The robot is created on start position which has been set in configuration file. Please see configuration file section for further information.
5. Type the controlling command in the command combo box, then click "send" to send out the command.

**Note:** This program has a simple AI for controlling spawned robots.

## 3. Communication (Message and Command)

The communication protocol used by server is TCP/IP. It opens a connection in war game engine and exchanges data with the outside.

After the robot is created on the simulator by connecting the client program to server, the client listens to the sensor data and receive the data, then player can sends commands to control the robot. To be assumed the robot equipped to 3 kinds of sensors, thus every client can receive the laser data and INS data and OPD (position and status of opponent robot) from server at per iteration and may enter control commands into server at any time.

### Sensor message:

- Laser {Data d1,d2,d3,...,d120}  
Where  
Laser sensor covers the 120 degree in front of robot.”d1, d2... d3” is a series of data values in meter.
- INS {Position x, y} {Rotation theta}

Where x, y, theta is a current location and orientation of own robot.

- OPD {Position x, y} {Rotation theta} {Life status}  
Where x, y is a position of opponent robot in meter.  
And theta is orientation of opponent robot in radian  
And life status is one of the following values:  
DEAD – opponent robot is dead  
LOW – opponent robot life is between 1-20 %  
MEDIUM – opponent robot life is between 21-50 %  
HIGH – opponent robot life is between 51-80 %  
VERYHIGH – opponent robot life is between 81-99 %  
FULL – opponent robot life is full (100%)
- STA {Life status}  
Where status is a value of own robot life in percent.
- Gift {Type gift} {Count c}  
Where gift is one of the following types:  
*STR, CRV, LIFE*  
C is a mount of each type.

Example:

Laser {Data

1.748221,1.749311,1.750939,1.753107,1.755819,1.759079,1.762892,1.767264,1.772201,1.7777  
12,1.783806,1.790492,1.79778,1.805684,1.814215,1.823389,1.833219,1.843725,1.854922,1.866  
832,1.879474,1.892874,1.907053,1.922039,1.937861,1.954549,1.972136,1.990659,2.010153,2.0  
30661,2.052227,2.074898,2.098725,2.123765,2.150074,2.177721,2.20677,2.2373,2.269391,2.30  
3133,2.33862,0.4706111,0.4619064,0.4536543,0.445826,0.4383954,0.4313383,0.4246326,0.418  
2586,0.4121974,0.406432,0.4009467,0.3957272,0.39076,0.3860325,0.3815336,0.3772526,0.373  
1796,0.3693054,0.3656217,0.3621205,0.3587947,0.3556373,0.3526419,0.3498029,0.3471147,0.  
3445722,0.3421706,0.3399057,0.3377734,0.3357698,0.3338916,0.3321353,0.3304981,0.328977  
2,0.3275701,0.3262745,0.3250883,0.3240095,0.3230365,0.3221676,0.3214016,0.3207372,0.320  
1734,0.3197093,0.3193443,0.3190776,0.3189089,0.3188381,0.3188649,0.3189894,0.3192117,0.  
3195324,0.3199517,0.3204705,0.3210894,0.3218094,0.3226317,0.3235576,0.3245884,0.325725  
9,0.326972,0.3283284,0.3297977,0.3313821,0.3330843,0.3349071,0.3368537,0.3389275,0.3411  
321,0.3434714,0.3459498,0.3485717,0.351342,0.3542661,0.3573497,0.3605987,0.3640199,0.36  
76203,0.3714075,0.3753898}

INS {Position -2.75,-0.01} {Rotation -2.0731}

OPD {Position -2.53,-0.05} {Rotation -1.57} {Life FULL}

STA {Life 100}

Gift {Type STR} {Count 5}

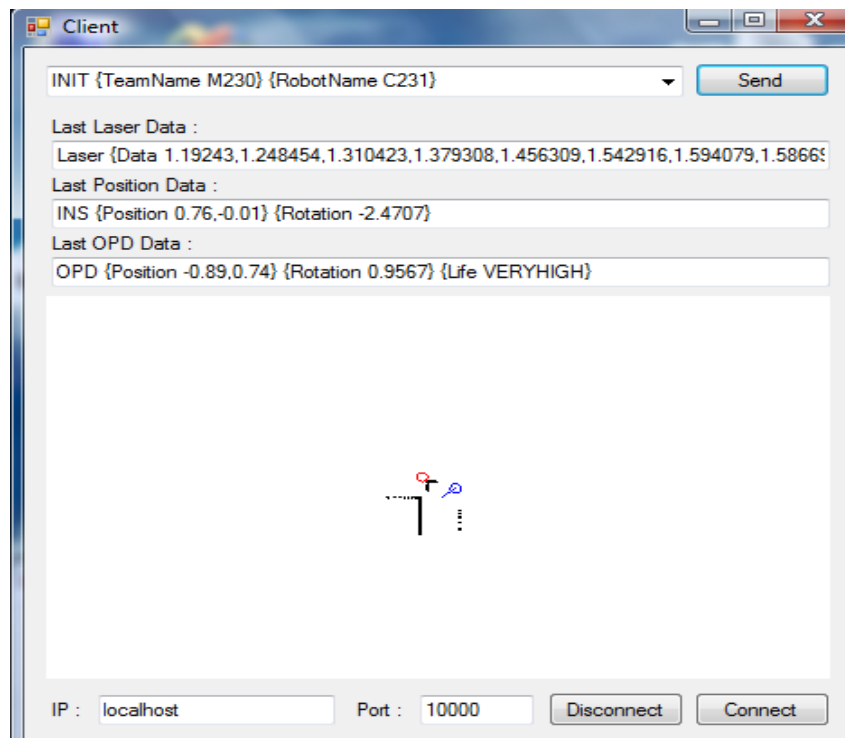


Figure 4: The Sample of Client program which has received laser and INS and OPD data

The client/server architecture makes it possible to add multiple robots into the simulator. Since every robot uses a socket to communicate, for every robot, the client must create a connection for it, so it is essential to edit the number player's parameter in a server configuration (server.cfg)

### Command Format:

There are currently 3 types of control command. The command format must be exactly similar to defined command. Name of commands isn't case sensitive but all tags like TeamName or Value is case sensitive.

- INIT Command

INIT {TeamName *string*} {RobotName *string*}

Where first "*string*" describes the name of team and second "*string*" describes the name of robot.

Example:

INIT {TeamName M230} {RobotName C1} add a robot to the world of game

- DRIVE Command

DRIVE {Type *string*} {Value *float*}

Where “*string*” referred to a type of drive Command. it will be one of the following values: ‘Forward’, ‘Rotate’, ‘Stop’ That this values aren’t case sensitive

“*float*” is movement value or rotation value . The value range is -1 to 1 and corresponds to the minimum and maximum linear velocity, respectively.

Example:

DRIVE {Type forward} {Value 0.1} will drive robot forward at a rate 0.1 meters per second.

DRIVE {Type forward} {Value -0.1} will drive robot backward at a rate of 0.1 meters per second.

DRIVE {Type rotate} {Value 1} will rotate robot at a rate of 20 degree per second.

DRIVE {Type stop} will stop robot at current position.

- FIRE Command

FIRE {Type *string*} {Angle *float*} {Speed *float*}

The robot was able to shoot to an opponent robot by sending a FIRE command. There are 3 input parameters for fire command.

Where “*string*” referred to a type of fire command it can be one the following values: ‘CRV’, ‘STR’ that this values aren’t case sensitive

*Angle “float”* is projectile angle around x axis in right handed 3d environment

*Speed “float”* is movement speed that it can be maximum 3 meter/sec

**Note:** You can’t use Angle and Speed parameters in STR type, because this type was implemented as a straight fire on robot direction.

The maximum number of fires for each type, that each robot can be allowed to use, has been set in sensor configuration file (sensors.cfg).

Example:

FIRE {Type CRV} {Angle 0.7875} {Speed 3}

FIRE {Type STR}

## 4. Sensor Simulation

Virtual War Game server1.0 includes 2 kinds of sensors. It will be extended to more sensors in new version of server.

- Position estimation sensor :These include position(x,y), orientation(theta)
- Perception sensors :These include laser sensor, Ins sensor, Fire sensor
  - Laser Sensor:

```
[Laser]
Fov=2.1
Res=0.0175
MinDist=0.15
MaxDist=4
DT=0.4
```

Where

Fov parameter is the scan field of view.

Res parameter is the scan resolution, the step length of rotating from start direction to the end direction.

MaxDis is a maximum distance that laser can detect.

MinDis is a minimum distance that laser can detect.

DT indicates the delta time in second.

- INS Sensor

```
[INS]
Noise=0
DT=0.2
```

Assuming that a noise of INS is zero.

- Fire Sensor

```
[Fire]
CRV_Count=5
CRV_MaxHit=30
CRV_LifeTime=12500
CRV_HitdownCoef=0.2
STR_Count=10
STR_MaxHit=10
STR_LifeTime=5000
```

```
STR_HitdownCoef=0.3  
DT=3
```

Where

Count indicates Maximum number of hit that a robot can use during the game.  
The value of MaxHit was used to decreased the power of robot due to robot got damaged with hitting.

## 5. Configuration files

This section contains three recommended configuration files in following:

- Recommended server configuration (server.cfg)

```
[Server]  
Port=10000  
  
#team settings  
[Team]  
NumPlayers=4  
NumTeams=2
```

The value of port determines which ports are in use by server and other client applications. The maximum allowed number of connections number has been determined with the number of team. If every team decided to utilize multi robot in the game, the default number of players could be change.

**Note:** You can change the parameters to the values that you want.

- Recommended robot configuration (robot.cfg)

```
[Robot]  
Model=robot.model  
InitialLaserPosition=0,75,0  
InitialRotation=0,0,0  
InitialScale=1,1,1  
Mass=50  
  
[Pose]  
Pose1=-100,0,-200,0,0,0  
Pose2=100,0,-200,0,-3.14,0
```

**Please note:** Pose values are (x, y, z)-(yaw, pitch, roll) in map. If you want to change the start position or need further position, you can change it with surrounding the given default.

- Map configuration (map.cfg)

In this file you can find the Bonus packs position in the map. This positions are static and determined by Technical mans. This position will be informed just before the run.

- Game configuration (game.cfg)

In this file you can find rules of some events like gift of achieving life bonus, fire bonus and penalty of non movable robots ( if your agent don't move or shoot during X time robot gets penalty )

Amount of X, sets in "ForceMoveTimeOut" tag in this file.

**Important Note:** These values are read-only and you can't change the values of these files during run



Figure 5: Simulation screenshot



Figure 6: Bonus pack screenshot