CS4039 MULTI-AGENT SYSTEMS
PROJECT REPORT

INFORMATION WARFARE SYSTEM

Submitted by:

ANKUR SINGH	B090800CS
AVIRAL NIGAM	B090871CS
DIMPI SAIKIA	B090613CS
MALI MUKESH	B090901CS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY - CALICUT

Index

Abstract	3
Introduction	4
Agents Model	5
Multi-agent Organization of the system	6
Types of Agents	7
Agent Architecture	8
The Contract net Protocol Model	9
State Transition Diagram	10
Conclusion	11
Tools/Framework and Languages	11
References	12
Java Code	12
Screenshots of the running code	22

Abstract

Military system simulations are usually used to train soldiers to perform missions and to learn to work together in teams and across command structures, or carry out the advanced concept technology demonstrations for operational applications of equipment systems on future battlefield.

It is a complex, dynamic, and information centric system with heterogeneous, autonomous members. To simulate its action, multi-agent based modeling technology is applied to set up the mappings from the members in IWS to respective agents, by which the distributed multi-agent system is designed. Thus, the multi-agent interactions centric platform-level virtual battlefield simulation system and its agent model are designed.

The established demonstration system proves the feasibility and efficiency of our model, and shows its advantages in realizing real time platform-level computer simulation for military systems.

The focus of this project is the MAS theory and its application in combat simulation.

Introduction

Multi-agent technology is a foreland of artificial intelligence (AI), especially in the field of distributed artificial intelligence (DAI). According to this technology, complex big system is decomposed into autonomous subsystems. With the communication and cooperation of these subsystems, the whole big system's some specific behavior and function emerged. The capturing of tactical warfare's realism, its interactions and unpredictability, is an issue needing to be studied.

Tactical warfare process has heterogeneous members, such as tanks, missile launch vehicles, armored reconnaissance vehicles, electronic reconnaissance platforms, and combat command platforms. These members have administrative levels and a lot of interactions, such as sending or receiving combat orders.
With the development and application of information and network technology, modern warfare is under a profound transformation over many aspects and the combat has more and more characteristics of a complex system.

An agent is viewed as an autonomous, internally motivated entity that is situated within a dynamic and not entirely predictable environment from which it receives perceptual inputs and to which it effects changes by performing actions.

We can think that tactical warfare system is in substance a distributed artificial intelligence system. Since an agent may have believes, desires, intentions, and it may adopt a role or have relationships with others, tactical warfare system can be looked upon as a collection of autonomous agents that are dependent upon each other. Therefore the method of modeling and simulation based on multi-agent interactions is applicable to our case. We design a multi-agent interactions centric platform level virtual battlefield simulation model to build underlying mechanisms for the advanced concept technology demonstration of warfare activities on future battlefield.

Agents Model

An intelligent agent with human being properties such as autonomy, sociality, adaptability and intelligence can act as a human. Agents may be seen as a natural extension of the concept of software objects. Agent-based programming adds abstraction entities, i.e. agents, which have an independent execution thread to the object-oriented paradigm. Thus, an agent is able to act in a goal- directed fashion.

A multi-agent system is a collection of agents co-operating with each other in order to fulfill common and individual goals where different agents often have different roles and individual goals.

An agent has four basic features:
(1) *Autonomy*. Autonomy has most often been defined as freedom from human intervention, oversight, or control. The agent can control the states and action with its own power.
(2) *Social ability*. The agent can communicate directly with the other agents.
(3) *Reactivity*. The agent can perceive the environment around it and react to the correlative events timely.
(4) *Positivity*. According to the goal of operation, agent will act actively and affect the outer environment to implement the object.

In the combat simulation multi-agent system, the agent should have the characteristics as follows:
(1) It is capable of acting in an environment.
(2) It can communicate directly with other agents.
(3) It is driven by a set of tendencies.
(4) It possesses resources of its own.
(5) It is capable of perceiving its environment (but to a limited extent).
(6) It has only a partial representation of this environment.
(7) It possesses skills and offer services.
(8) It may be able to reproduce itself.

(9) Its behavior tends towards satisfying its objectives.

There are some characteristics for an ideal application of agent technology:
Modular: In the sense that each entity has a well defined set of state variables that is distinct from those of its environment and that the interface to the environment can be clearly identified.

Decentralized: In the sense that the application can be decomposed into stand-alone software processes capable of performing useful tasks without continuous direction from some other software process.

Changeable: In the sense that the structure of the application may change quickly and frequently. Ill-structured: In the sense that all information about the application is not available when the system is being designed.

Complex: In the sense that the system exhibits a large number of different behaviors that may interact in sophisticated ways.

Multi-agent Organization of System

In our project, we design a multi-agent interactions centric platform-level virtual battlefield simulation model to build underlying mechanisms of warfare activities on future battlefield. The model is presented to support multi-agent interactions centric simulation by using an improved Contract Net Protocol, where state transitions are studied to describe the interactions. Based on the analysis on the requirement and counter- measure, the mapping from tactical warfare system's members to respective intelligent agents is set up. Since an agent may have beliefs, desires, intentions, and it may adopt a role or have relationships with others, tactical warfare system impersonates a collection of autonomous agents that are dependent upon each other. Agents may be seen as a natural extension of the concept of software objects.

Types of Agents

Information Warfare System (IWS) has heterogeneous members, such as intelligence reconnaissance platforms, digital armored fight platforms, electronic warfare platforms, digital fire support and logistic support platforms, which have lives and administrative levels.

The function agents in Red force include tank agents (TA), photo-reconnaissance vehicle agents (PRVA), radar reconnaissance vehicle agents (RRVA), armored reconnaissance vehicle agents (ARVA), cannon agents (CA), combat command vehicle agent (CCVA) and logistic support platform agents (LSPA). They are aggregated into the Red agent's federation.

The function agents in Blue force are similar to those Red force agents, but some different agents, e.g., armored cavalry vehicle agents (ACVA), missile launch vehicle agents (MLVA), trench mortar agents (TMA) and information processing vehicle agents (IPVA) are designed. Since there are some differences in force organization, they are aggregated into the Blue agent's federation.

The administration agents and service agents include federation manager agent, declare manager agent, time manager agent, data distribution manager agent, and so on, which play the roles of demonstration control (DC), simulation evaluation (SE), data base (DB), situation displaying (SD), command practice (CP) and battlefield environment (BE). These agents can be aggregated into the "White" federation.

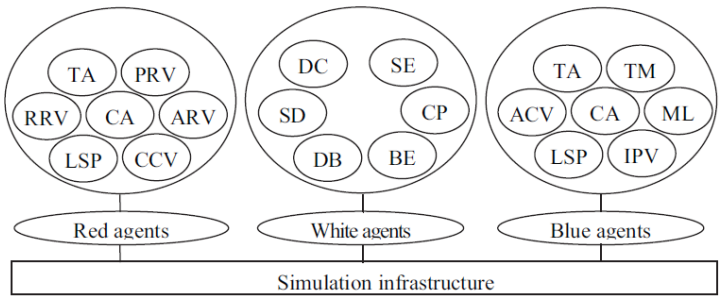


Figure 1. Multi-agent battlefield entities simulation system architecture.

Agent Architecture

The architecture of MAS consists of the models of every agent in the system and the interactions between them. The MAS uses a bottom up modeling technique to capturing the interactions taking place between the system's constituent units. Agent has the attributes of autonomy and cooperation. The autonomy attribute makes the agent adapt the dynamic complex environment, while the cooperation attribute makes up the deficiency of signal agent.

The main components are:
Behavior: It is responsible for controlling the other components of the agent.
Collaborator: It controls the interactions with the other agents.
Sensor: It is the gateway to the perceptions of the agent about the external environment
Effectors: It produces changes in the external environment.
Communicator: It is a component of the Collaborator that completely hides the medium used for exchanging messages.

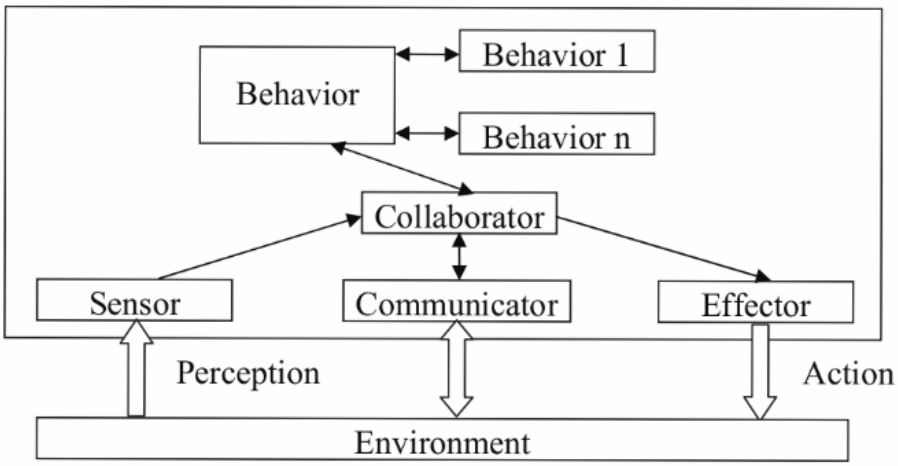


Figure 2. Internal model of agent.

The Contract net Protocol Model

Contract Net Protocol proposes episodic rounds of inter-communication acts. In our case, tactical warfare system consists of a Red armored force unit (one combat command vehicle, nine tanks and some armored reconnaissance platforms) and a Blue army troop (one information processing vehicle, one tank, one missile launch vehicle, one trench mortar, and some other fire platforms).

The roles: Initiator – Manager
Contractors – Other entity agents

The manager issues the call for proposals, and other interested agents send proposals.
The proposals are collected by the manager, and then they are refused or accepted. The accepted proposals may be cancelled sometimes. Then the manager returns a “failure” or “success” message.

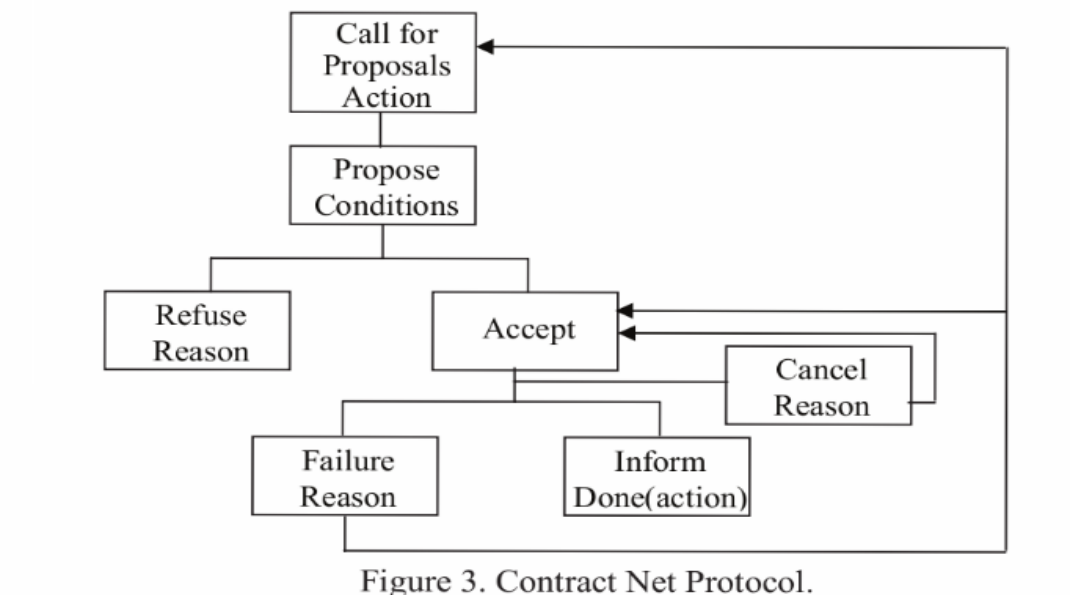


Figure 3. Contract Net Protocol.

State Transition Diagram

The entity agent interactions protocol's denominations:
The orientation of an arrow represents the message flow direction of an entity agent.
“Recv” means that a message is received correctly.
“Send” means that a message is sent correctly. Recv or Send follows the type of a message;
“Refused” and “Accepted” represents respectively the agent interaction results, i.e., success and failure.

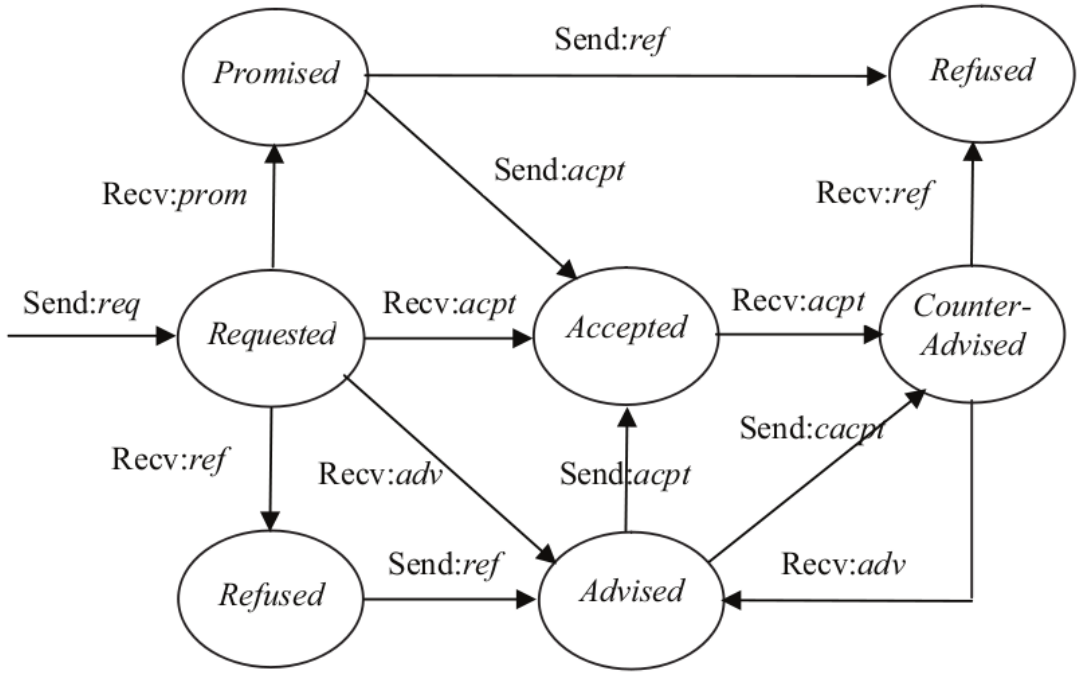


Figure 4. State transition diagram of entity agent interactions protocol.

Conclusion

In order to solve the problems about simulation of intelligent engagement action, dynamic information processing, and changing battle space environment, we analyzed the problem space, battle space and information processing of IWS, put forward the multi-agent organization of IWS and the architecture of agents, and furthermore set up the elementary approach model task decomposition and task allocation in this system.

A multi-agent interactions centric platform-level virtual battlefield simulation model is designed. The multi-agent organization of platform level simulation system and the architecture of entity agents are put forward. In this system, an improved Contract Net Protocol is used for the entity agent interactions model. This model can be used to understand the external, complicated and intelligent tactical warfare resources application and can realize the dynamic platform-level battlefield activities simulation.

The established distributed multi-agent system model of IWS can afford advantage for carrying out the advanced concept technology demonstration of the external, dynamic, complicated and intelligent process.

Tools/Framework

- Java Agent Development Framework (JADE)
- MaDKit

Languages

- JAVA
- Agent UML

References

[1] Multi-Agent Model of Information Warfare System.
Xiong Li; Jianhua Luo; Yukun Cao

[2] Multi-Agent interactions centric virtual battlefield simulation model.
Hongwei An; Xiong Li; Xiuquan Xie

Java Code

/* Blue Agents */

```
package information.warfare;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Image;
import java.awt.Toolkit;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import madkit.gui.OutputPanel;
import madkit.kernel.Agent;
import madkit.kernel.Madkit;
import madkit.kernel.Madkit.LevelOption;
import madkit.kernel.Madkit.Option;
import madkit.kernel.Message;
import madkit.message.ObjectMessage;


/**
 *
 * @author aviral
 *
 */

@SuppressWarnings("serial")
```

```
public class BlueAgents extends Agent
{

     static int nbOfBlueAgentsOnScreen = 0;
     private JPanel blinkPanel;
     private static ImageIcon blueArmy = new ImageIcon(new
ImageIcon(BlueAgents.class.getResource("images/client.png")).getI
mage().getScaledInstance(70, 70, Image.SCALE_SMOOTH));
     private String weapon =
WhiteAgents.availableWeapons.get((int)
(Math.random()*WhiteAgents.availableWeapons.size()));

     @Override
     protected void activate()
     {
          createGroupIfAbsent("war","war-red-agents",true,null);
          requestRole("war","war-red-agents", "blueagent",null);
          int pause = 1000 + (int) (Math.random()*2000);
          if(logger != null)
               logger.info("I will be looking for a "+weapon+" in
"+pause+" ms !");
          pause(pause);
     }

     @SuppressWarnings("unchecked")
     @Override
     protected void live()
     {
          boolean haveWeapon = false;

          while (! haveWeapon)
          {
               Message commanderOrders = null;
               while (commanderOrders == null) {
                         sendMessageWithRole(
                                   "war",
                                   "war-red-agents",
                                   "commander",
                                   new
ObjectMessage<String>(weapon),
                                   "blueagent");
                         if(logger != null)
                              logger.info("Waiting for commander
to answer...");
                         commanderOrders = waitNextMessage(1000);
                         if(logger != null && commanderOrders ==
null)
                              logger.info("For now there is
nothing for me :(");
```

```
                    }
                takeOrders(commanderOrders);// I found a commander
and he found something for me
                haveWeapon = takeWeapon((ObjectMessage<String>)
commanderOrders);
                }
          }

     @Override
     protected void end() {
          if(logger != null)
               logger.info("I have been hit, i m about to die
!!!");
          pause((int) (Math.random() * 2000 + 500));
          launchAgent(new BlueAgents(),4,true);
     }

     private void takeOrders(Message commanderReply) {
          if(logger != null)
               logger.info("I found a commander :
"+commanderReply.getSender());
          if (blinkPanel != null) {
               blinkPanel.setBackground(Color.BLUE);
               pause(1000);
          }
     }

     private boolean takeWeapon(ObjectMessage<String>
commanderAnswer) {
          String regiment = commanderAnswer.getContent();
          requestRole("war",regiment, "blueagent");
          Message ticket = sendMessageAndWaitForReply("war",
regiment, "redagent",new ObjectMessage<String>("money"), 4000);
          if(ticket != null){
               if(logger != null)
                    logger.info("Yeeeaah: I have my ticket :) ");
               if (hasGUI()) {
                    blinkPanel.setBackground(Color.BLUE);
               }
               leaveGroup("war", "war-red-agents");
               pause((int) (1000+Math.random()*2000));
               return true;
          }
          return false;

     }

     @Override
     public void setupFrame(JFrame frame) {
```

```
          JPanel p = new JPanel(new BorderLayout());
          //Customising but still using the OutputPanel from
MaDKit GUI
          p.add(new OutputPanel(this),BorderLayout.CENTER);
          blinkPanel = new JPanel();
          blinkPanel.add(new JLabel(blueArmy));
          blinkPanel.add(new JLabel(new
ImageIcon(getClass().getResource("images/"+weapon+".png"))));
          p.add(blinkPanel,BorderLayout.NORTH);
          p.validate();
          frame.add(p);
          int xLocation = nbOfBlueAgentsOnScreen++*390;
          if(xLocation+390 >
Toolkit.getDefaultToolkit().getScreenSize().getWidth())
               nbOfBlueAgentsOnScreen=0;
          frame.setLocation(xLocation, 0);
          frame.setSize(390, 300);
     }

     /**
      * @param args
      */
     public static void main(String[] args) {
          nbOfBlueAgentsOnScreen=0;
          WhiteAgents.nbOfWhiteAgentsOnScreen=0;
          String[] argss = {
                    LevelOption.agentLogLevel.toString(),"INFO",
                    LevelOption.kernelLogLevel.toString(),"OFF",
                    Option.launchAgents.toString(),

BlueAgents.class.getName()+",true,2;"+

WhiteAgents.class.getName()+",true,3;"+

RedAgents.class.getName()+",true,7"
          };
          Madkit.main(argss);
     }
}
```

/* Red Agents */

```
package information.warfare;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Toolkit;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import madkit.gui.OutputPanel;
import madkit.kernel.Agent;
import madkit.kernel.Message;
import madkit.message.ObjectMessage;

/**
 *
 * @author aviral
 *
 */

public class RedAgents extends Agent
{
	private static final long serialVersionUID = 1L;
	private String competence;
	static private int nbOfRedAgentsOnScreen=0;
	private JPanel blinkPanel;


	public RedAgents()
	{
		competence = WhiteAgents.availableWeapons.get((int)
(Math.random()*WhiteAgents.availableWeapons.size()));
	}

	public void activate()
	{
		createGroupIfAbsent("war","war-blue-agents",true,null);
		requestRole("war","war-blue-agents",competence+"-
redagent",null);
	}


	@SuppressWarnings("unchecked")
	public void live()
	{
```

```
          while (true) {
               Message m = waitNextMessage();
               if (m.getSender().getRole().equals("commander"))
                    whiteAgentMessage((ObjectMessage<String>) m);
               else
                    finalizeWeapon((ObjectMessage<String>) m);
          }
     }

     private void whiteAgentMessage(ObjectMessage<String> m) {
          if(m.getContent().equals("make-bid-please")){
               if(logger != null)
                    logger.info("I received a call for bid from
"+m.getSender());
               sendReply(m, new ObjectMessage<Integer>((int)
(Math.random()*500)));
          }
          else{
               weaponSelected(m);
          }
     }

     private void weaponSelected(ObjectMessage<String> m) {
          if (hasGUI()) {
               blinkPanel.setBackground(Color.RED);
          }
          if(logger != null)
               logger.info("I have been selected :)");
          String regiment = m.getContent();
          createGroup("war", regiment,true);
          requestRole("war", regiment, "redagent");
          sendReply(m, new Message()); // just an acknowledgement
     }

     private void finalizeWeapon(ObjectMessage<String> m) {
          if (hasGUI()) {
          blinkPanel.setBackground(Color.RED);
          }
          if(logger != null)
               logger.info("I have sold something: That's great
!");
          sendReply(m, new ObjectMessage<String>("ticket"));
          pause((int) (Math.random()*2000+1000));//let us
celebrate !!
          leaveGroup("war", m.getSender().getGroup());
          if (hasGUI()) {
               blinkPanel.setBackground(Color.RED);
          }
     }
```

```
     @Override
     public void setupFrame(JFrame frame) {
          JPanel p = new JPanel(new BorderLayout());
          //Customising but still using the OutputPanel from
MaDKit GUI
          p.add(new OutputPanel(this),BorderLayout.CENTER);
          blinkPanel = new JPanel();
          blinkPanel.add(new JLabel(new
ImageIcon(getClass().getResource("images/"+competence+".png"))));
          p.add(blinkPanel,BorderLayout.NORTH);
          blinkPanel.setBackground(Color.RED);
          p.validate();
          frame.add(p);
          int xLocation = nbOfRedAgentsOnScreen++*300;
          if(xLocation+300 >
Toolkit.getDefaultToolkit().getScreenSize().getWidth())
               nbOfRedAgentsOnScreen=0;
          frame.setLocation(xLocation, 640);
          frame.setSize(300, 300);
     }
}
```

/* White Agents */

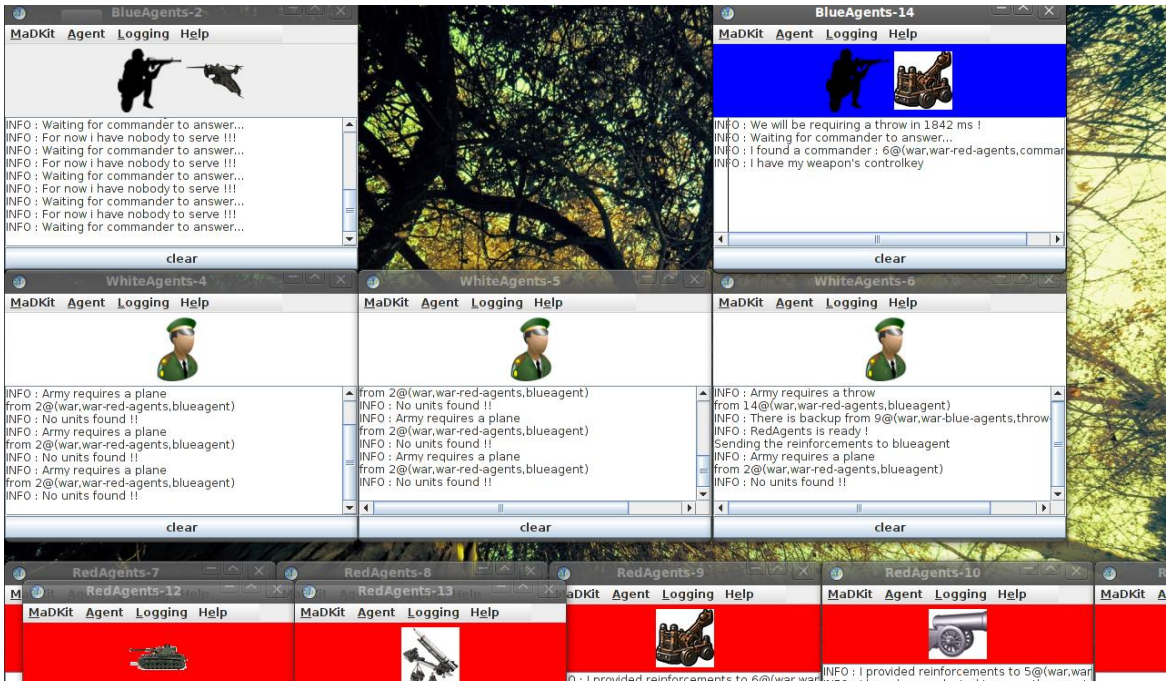
```
package information.warfare;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Image;
import java.awt.Toolkit;
import java.util.Arrays;
import java.util.List;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import madkit.gui.OutputPanel;
import madkit.kernel.Agent;
import madkit.kernel.Message;
import madkit.message.ObjectMessage;

/**
 *
 * @author aviral
 *
 */

public class WhiteAgents extends Agent
{
     private static final long serialVersionUID = 1L;
     static int nbOfWhiteAgentsOnScreen=0;
     private JPanel blinkPanel;
     public static List<String> availableWeapons =
Arrays.asList("tank","plane","boat");
     private static ImageIcon commanderImage= new ImageIcon(new
ImageIcon(BlueAgents.class.getResource("images/broker.png")).getI
mage().getScaledInstance(70, 70, Image.SCALE_SMOOTH));

     public void activate()
     {
          createGroupIfAbsent("war","war-red-agents",true,null);
          createGroupIfAbsent("war","war-blue-agents",true,null);
          requestRole("war","war-blue-agents", "commander",null);
          requestRole("war","war-red-agents", "commander",null);
     }


     @SuppressWarnings("unchecked")
```

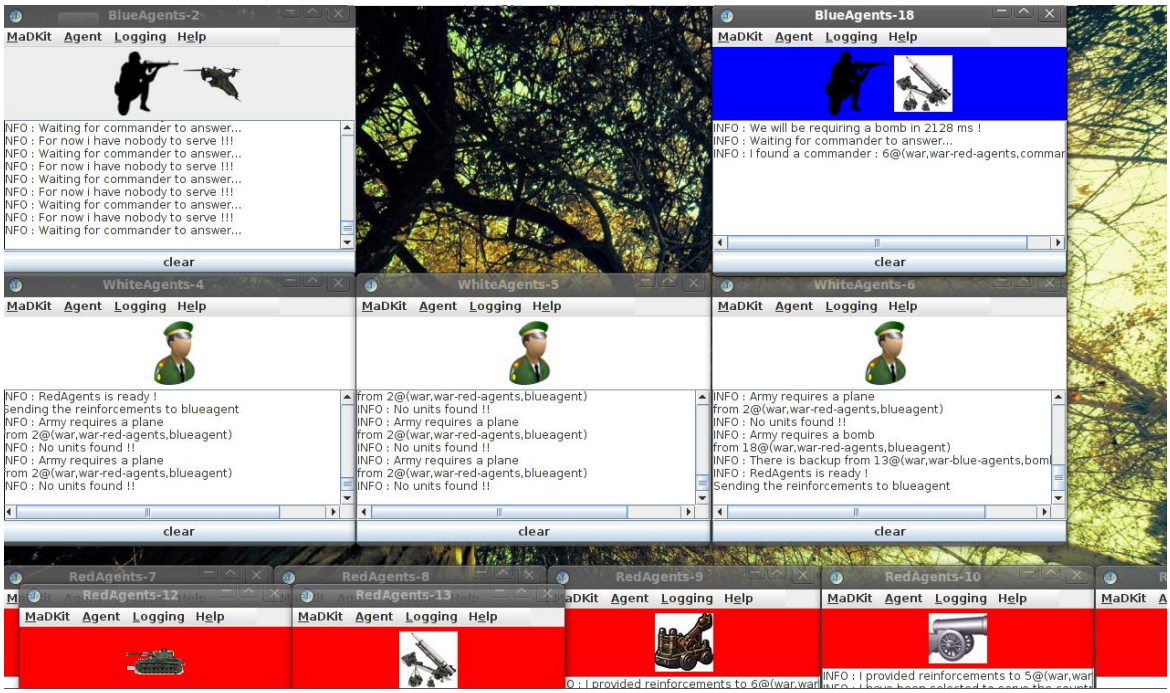
```
	public void live()
	{
		while (true)
		{
			Message m;
			m = purgeMailbox();
			if (m == null) {
				m = waitNextMessage();
			}
			String role = m.getSender().getRole();
			if(role.equals("blueagent")){
				blueAgentRequest((ObjectMessage<String>) m);
			}
		}
	}

	private void blueAgentRequest(ObjectMessage<String> request)
{
		if(! request.getSender().exists())
			return;
		if (hasGUI()) {
			blinkPanel.setBackground(Color.WHITE);
		}
		if(logger != null)
			logger.info("I received a request for a
"+request.getContent()+" \nfrom "+request.getSender());
		List<Message> technology =
broadcastMessageWithRoleAndWaitForReplies(
				"war",
				"war-blue-agents",
				request.getContent()+"-redagent",
				new ObjectMessage<String>("make-bid-please"),
				"commander",
				900);
		if(technology == null){
			if(logger != null)
				logger.info("No bids at all !!");
			if (hasGUI()) {
				blinkPanel.setBackground(Color.WHITE);
			}
			return;
		}
		Message m = selectBestWeapon(technology);
		if (m != null) {
			if(logger != null)
				logger.info("There is one interesting offer
from "+m.getSender());
			String regiment = ""+System.nanoTime();
			Message ack = sendMessageWithRoleAndWaitForReply(
```

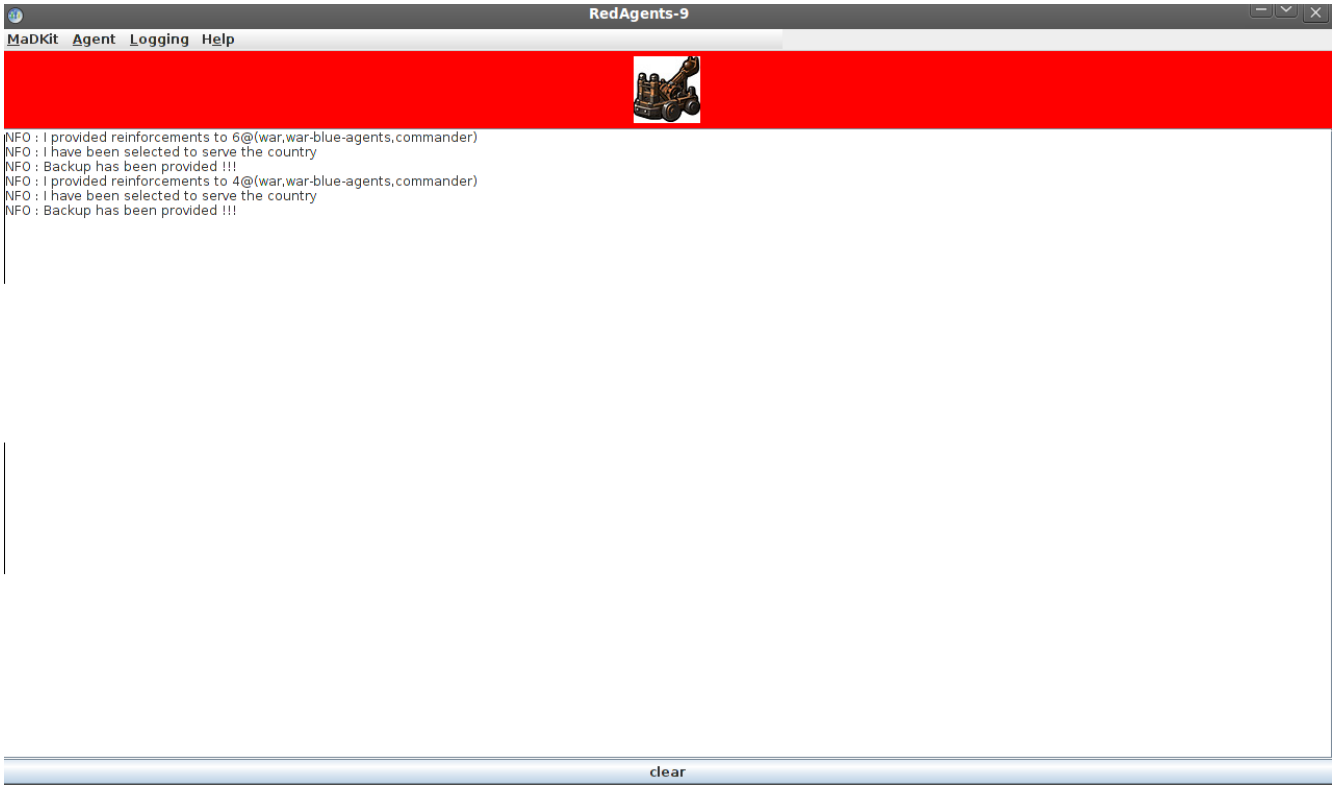
```
                        m.getSender(),
                        new ObjectMessage<String>(regiment),
                        "commander",
                        1000);
                if(ack == null){
                    if(logger != null)
                        logger.info("RedAgents disappears !!");
                    return;
                }
                if(logger != null)
                    logger.info("RedAgents is ready !\nSending
the contract number to blueagent");
                sendReply(request, new
ObjectMessage<String>(regiment));
                pause((int) (Math.random()*2000+1000));//let us
celebrate !!
            }
            if (hasGUI()) {
                blinkPanel.setBackground(Color.WHITE);
            }
        }

        @SuppressWarnings("unchecked")
        private Message selectBestWeapon(List<Message> bids) {
            ObjectMessage<Integer> best = (ObjectMessage<Integer>)
bids.get(0);
            for(Message m : bids){
                ObjectMessage<Integer> offer =
(ObjectMessage<Integer>) m;
                if(best.getContent() > offer.getContent()){
                    best = offer;
                }
            }
            return best;
        }

        @Override
        public void setupFrame(JFrame frame) {
            JPanel p = new JPanel(new BorderLayout());
            //Customising but still using the OutputPanel from
MaDKit GUI
            p.add(new OutputPanel(this),BorderLayout.CENTER);
            blinkPanel = new JPanel();
            blinkPanel.add(new JLabel(commanderImage));
            p.add(blinkPanel,BorderLayout.NORTH);
            blinkPanel.setBackground(Color.WHITE);
            p.validate();
            frame.add(p);
            int xLocation = nbOfWhiteAgentsOnScreen++*390;
```

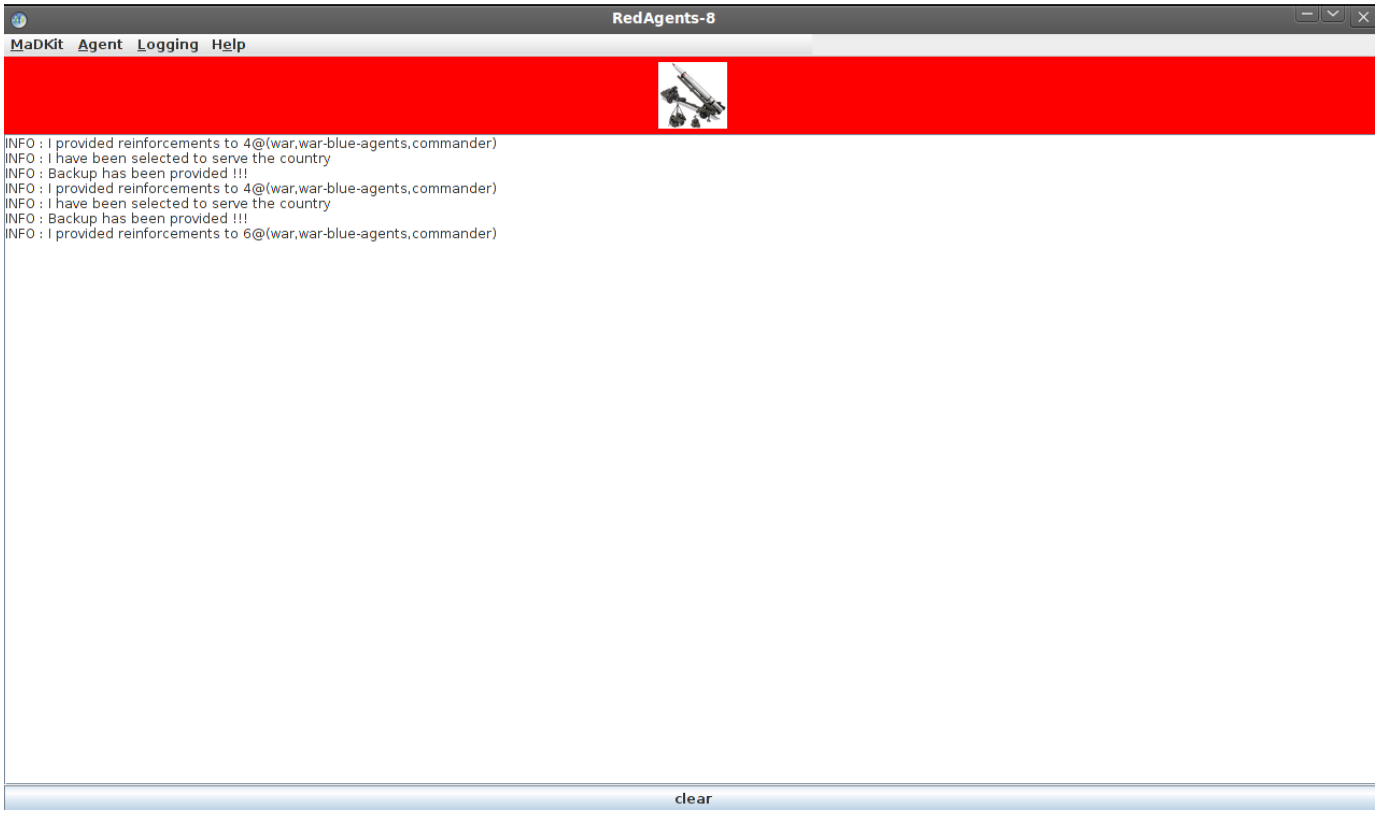
```
            if(xLocation+390 >
Toolkit.getDefaultToolkit().getScreenSize().getWidth())
                  nbOfWhiteAgentsOnScreen=0;
            frame.setLocation(xLocation, 320);
            frame.setSize(390, 300);
      }
}
```

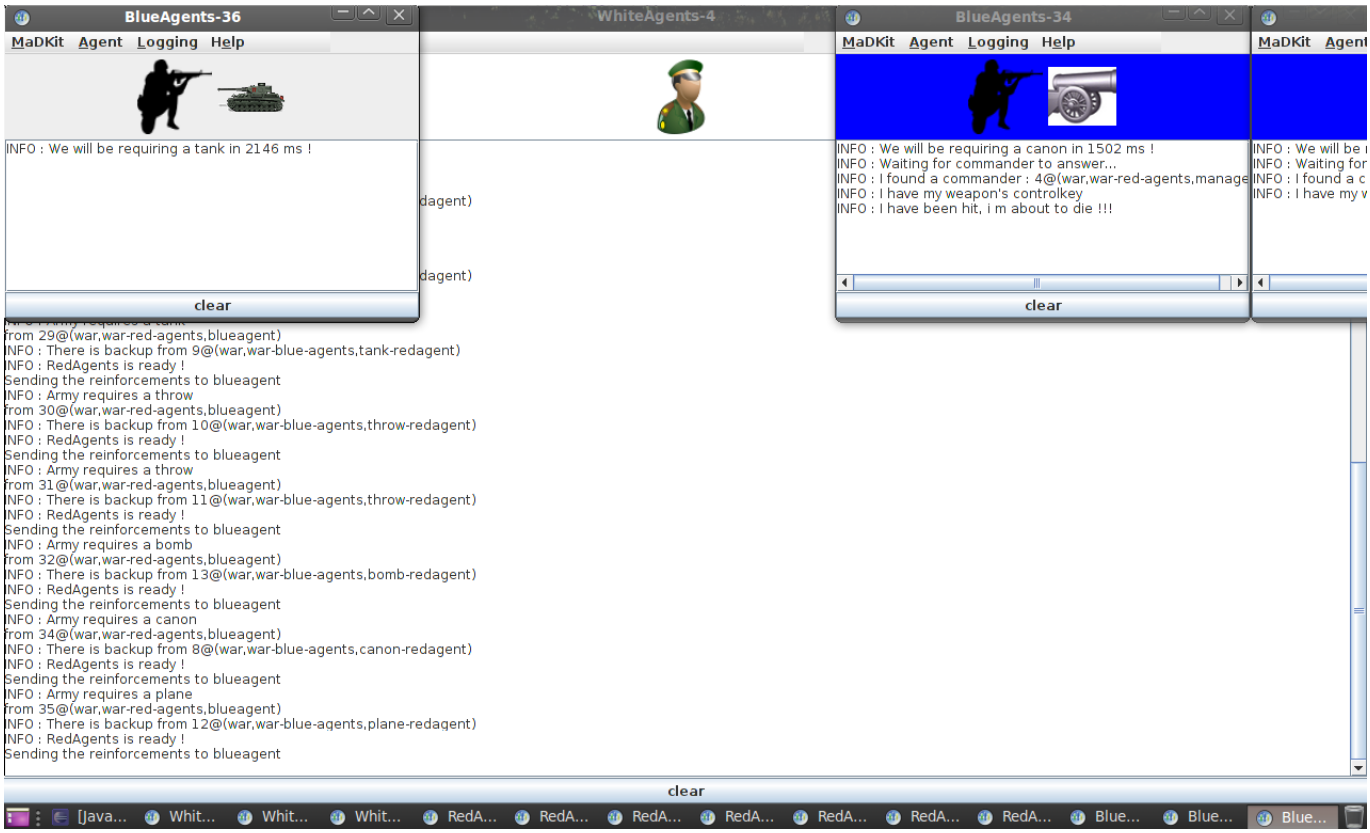
Screenshots of the running code

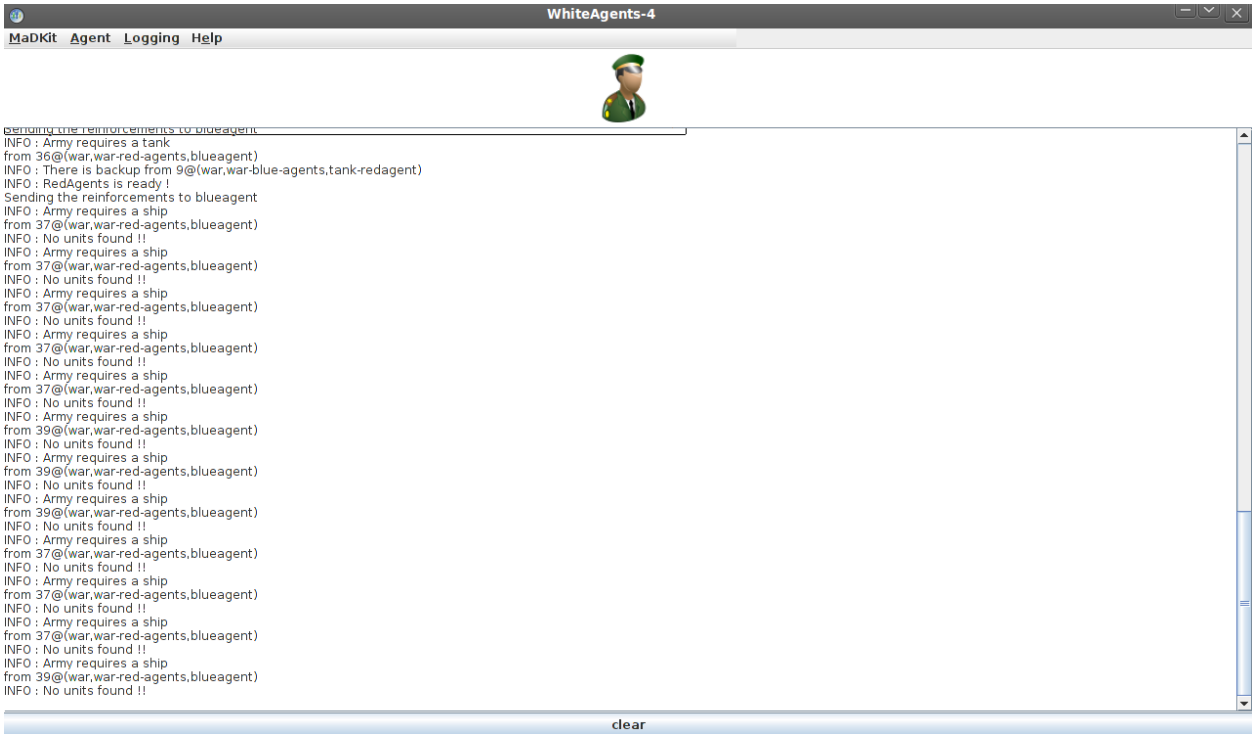


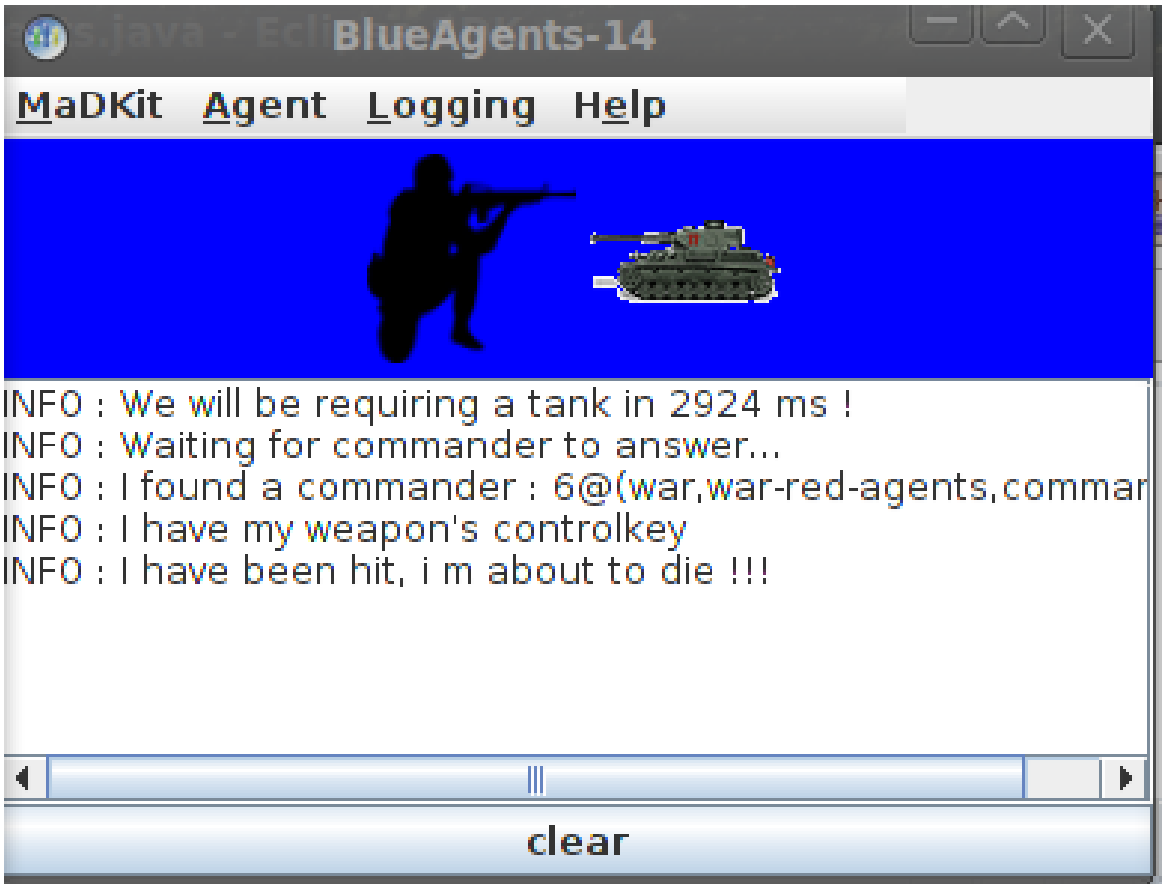



RedAgents-9
MaDKit Agent Logging Help
NFO : I provided reinforcements to 6@(war,war-blue-agents,commander)
NFO : I have been selected to serve the country
NFO : Backup has been provided !!!
NFO : I provided reinforcements to 4@(war,war-blue-agents,commander)
NFO : I have been selected to serve the country
NFO : Backup has been provided !!!
clear


RedAgents-8
MaDKit Agent Logging Help
INFO : I provided reinforcements to 4@(war,war-blue-agents,commander)
INFO : I have been selected to serve the country
INFO : Backup has been provided !!!
INFO : I provided reinforcements to 4@(war,war-blue-agents,commander)
INFO : I have been selected to serve the country
INFO : Backup has been provided !!!
INFO : I provided reinforcements to 6@(war,war-blue-agents,commander)
clear


BlueAgents-36
MaDKit Agent Logging Help
INFO : We will be requiring a tank in 2146 ms !
clear
WhiteAgents-4
BlueAgents-34
MaDKit Agent Logging Help
INFO : We will be requiring a canon in 1502 ms !
INFO : Waiting for commander to answer...
INFO : I found a commander : 4@(war,war-red-agents,manage
INFO : I have my weapon's controlkey
INFO : I have been hit, i m about to die !!!
clear
from 29@(war,war-red-agents,blueagent)
INFO : There is backup from 9@(war,war-blue-agents,tank-redagent)
INFO : RedAgents is ready !
Sending the reinforcements to blueagent
INFO : Army requires a throw
from 30@(war,war-red-agents,blueagent)
INFO : There is backup from 10@(war,war-blue-agents,throw-redagent)
INFO : RedAgents is ready !
Sending the reinforcements to blueagent
INFO : Army requires a throw
from 31@(war,war-red-agents,blueagent)
INFO : There is backup from 11@(war,war-blue-agents,throw-redagent)
INFO : RedAgents is ready !
Sending the reinforcements to blueagent
INFO : Army requires a bomb
from 32@(war,war-red-agents,blueagent)
INFO : There is backup from 13@(war,war-blue-agents,bomb-redagent)
INFO : RedAgents is ready !
Sending the reinforcements to blueagent
INFO : Army requires a canon
from 34@(war,war-red-agents,blueagent)
INFO : There is backup from 8@(war,war-blue-agents,canon-redagent)
INFO : RedAgents is ready !
Sending the reinforcements to blueagent
INFO : Army requires a plane
from 35@(war,war-red-agents,blueagent)
INFO : There is backup from 12@(war,war-blue-agents,plane-redagent)
INFO : RedAgents is ready !
Sending the reinforcements to blueagent
clear


WhiteAgents-4
MaDKit Agent Logging Help
INFO : Army requires a tank
from 36@(war,war-red-agents,blueagent)
INFO : There is backup from 9@(war,war-blue-agents,tank-redagent)
INFO : RedAgents is ready !
Sending the reinforcements to blueagent
INFO : Army requires a ship
from 37@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 37@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 37@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 37@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 37@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 39@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 39@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 39@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 37@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 37@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 37@(war,war-red-agents,blueagent)
INFO : No units found !!
INFO : Army requires a ship
from 39@(war,war-red-agents,blueagent)
INFO : No units found !!
clear


BlueAgents-14
MaDKit Agent Logging Help
INFO : We will be requiring a tank in 2924 ms !
INFO : Waiting for commander to answer...
INFO : I found a commander : 6@(war,war-red-agents,commar
INFO : I have my weapon's controlkey
INFO : I have been hit, i m about to die !!!
clear