Interactive Simulation in Crisis Management

T. Benjamins
Man-Machine Interaction Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
The Netherlands
T.Benjamins@ewi.tudelft.nl

L.J.M. Rothkrantz
Man-Machine Interaction Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
The Netherlands
L.J.M.Rothkrantz@ewi.tudelft.nl

ABSTRACT

Experiments in crisis management are expensive and difficult to realize. There is also a lack of training facilities in real crisis environments. Serious games and simulation can provide an alternative. We developed a system which enables interactive simulation for crisis management. It is called IMACSIM (Interactive Multi Agent Crisis Simulator Interpreter and Monitor). It is composed of the following components: First a software based platform for dynamic simulating of disasters. Next an event generator which can generate different crises situations. We designed a communication infrastructure that allows agents participants in the simulation to exchange messages. Every agent is able to observe the results of crisis events, process these events and initiate appropriate actions via a waypoint system. The decision making process is distributed among autonomous agents. Some actions may have an impact on the event generator, so there is an interaction between agents and event generator. We developed a first prototype. The design and test results will be described in this paper.

Keywords

Interactive simulation, serious gaming, crisis management, multi agent systems

INTRODUCTION

In recent years we observe an enormous growth in the scale and complexity of the terrorist attacks and natural disasters. It proves that the existing infrastructure and crisis management is unable to face the challenges of such events. There is a need for additional research, training methods and training facilities. But to set up a test in real life crisis situation is far from trivial. To research for example communication and corresponding infrastructure during terrorist attacks or flooding is infeasible. Software from the game industry can be applied to crisis context, to provide a virtual simulation environment for research or an interactive method for training in the field of emergency response.

For many years disasters are simulated for training and research. Disasters are generated using a fixed script, based on an XML file. In serious gaming dynamics scripts are used. Serious games are interactive. Users can change the order of the crisis events and the environment has to be adapted taking care of the reactions of the user.

Recently the COMBINED (Chaotic Open world Multi-agent Based Intelligently NEtworked Decision-support Systems) project (Combined, 2006) was finished, which was an initiative of DECIS-lab, in combination with several Dutch research partners, including TU Delft, The university of Amsterdam, TNO, Thales and several Fire departments (NIFV). The goal of this project was to design systems that consist of human actors and artificial agents. These agents work together to achieve their common goals, even if this means they are functioning in chaotic circumstances. Examples of other work in this field can be found in (Tatomir et al, 2006) and (Klapwijk et al, 2006).

Our research conducted for IMACSIM fits in the framework of the COMBINED project and had three main goals: to create a crisis environment in which different crises can be simulated by means of an event generator, to create a communication layer through which agents can exchange messages, and to create intelligent software agents that move around in the crisis world and their actions can change the events in the world. Finally users can play the role of agents so there is a mix of human and artificial agents.

At this moment game software is available which can be used for serious gaming. This software provides tools to build a realistic environment and to design bots with realistic behavior. We decided to develop our own software. First software tools are not freely available. Games which are able to generate realistic crisis events are not well developed. Next our focus was how to design the communication between agents and to model agents. In future implementation iterations gaming software will be used to design 3D-graphic visualization to represent the data currently available in side the simulation world.

This paper will have the following structure: In the next section we will introduce related work and our World model for IMACSIM. Then the simulator will be discussed. The next section Implementation will be dedicated to introducing the designed software components and the way external software components like JADE and Jess was included. In the Evaluation section an example of a testing scenario is given and our user tests, and finally we end this paper with a conclusion.

RELATED WORK

At this moment we observe an explosion of serious games initiatives, tools and games. Many of them are developed in the military domain. "America's Army" and Darwars "Ambush" are on of the most popular serious games. Another example is "Incident Commander" which teaches incident management for multiple scenario's, including terrorist attacks, and natural disasters. Both games are built on the top of other games, using the game engine. "America's Army" is built on top of "Unreal Tournament" engine" and Darwars "Ambush" on top of Operation Flash Point. In this section we will further restrict ourselves to the simulation and games which are on the basis of our research.

-In 2005 some authors present interesting projects and ideas on the Winter Simulation Conference. We mention "a simulation based training tool" for incident management (Jain, 2005), "a model of an emergency operations center with agents" (Loper 2005) and finally "a tool for first responder information flow simulations" (Robinson, 2005).

-The US government organizations EPA (Environmental Protection Agency) and NOAA (National Oceanic and Atmospheric Administration) have made a very easy-to-use gas dispersion modeling tool called ALOHA (Arial Locations Of Hazardous Atmospheres) (ALOHA,2004). It is able to model chemical releases and has some advanced features in modeling. ALOHA has been designed for people with crisis response duties, so after an incident they can get a fast overview of what is going on and what will be the situation in the upcoming hour.

-Hazmat: Hotzone is a simulation that uses video game visualization techniques to train first responders about how to respond to emergencies concerning hazardous materials. It is currently in the development stage at the Entertainment Technology Center at Carnegie Mellon University in collaboration with the Fire Department of New York. Hazmat: Hotzone is still in development and is estimated to be completed in Spring 2007 (HAZMAT, 2006).

-Another project focused on crisis situations is the Rescue project. The drillsim simulator, which is a part of the CAMAS (Crisis assessment, mitigation and analysis)-test bed (Mehrotra, 2005) is a multi-agent crisis simulator in which crisis response roles (e.g., evacuation) can be played by game participants. The simulator models different response activities at both tactical and operational levels, and model the information flow between different crisis response agencies. External components can be inserted at different interfaces between these activities or at some point of the information flow in order to study the effectiveness of research solutions in disaster management and tested for utility in disaster response. In addition this simulator can integrate real life drills into a digital simulation.

MODEL

To describe the concepts in IMACSIM, we have devised a World Model that is divided into three parts (see Figure 1), being the Real World, the Observed World and the Crisis Center. Through these parts of the world model we will explain the IMACSIM concepts.

We designed a model of the real world good enough to fit our needs. In this world there are a lot of waypoints. Waypoints are considered to be data storage with information about the area directly surrounding a certain point. This data can also be updated. These waypoints contain a lot of data that the agents walking through the area use to gather information about the crisis at hand. So agents are able to sense the world via waypoints, they read the data in the nearest waypoint. Those waypoints are located all through the area. The main reason we use waypoints is because of the reduction of computational complexity that is achieved when we only have to calculate physical properties for a limited number of points in the world instead of for every possible coordinate in the entire area. As long as the computations for a certain point are reasonably accurate it will serve our purpose very well. While the

simulation lasts, the physical properties of a waypoint are subject to change. In our first demonstrator waypoints are located on a grid at fixed distances.

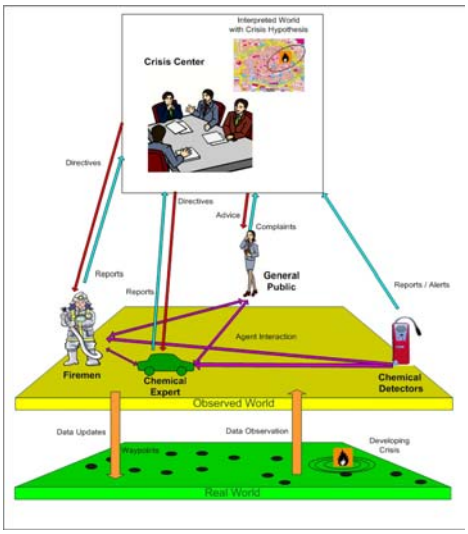

Figure 1. Global Overview of IMACSIM

In the Observed world all the agents see the events unfold and report about it to the Crisis Center or to other agents. In Figure 1 we can see that the different agents can report things to each other. Also they can complain or report observed effects of events to the crisis center. In the Observed World operate all different types of agents that have one thing in common: They observe the world. But because they all have a different perspective on the world, they interpret the effect of certain event differently, and therefore also react differently to those effects. This leaves room for flexible interpretation of situations that in real life is one of the key characteristics of crisis response. In the observed world operate different types of agents that all have their own view of the situation.

All information acquired in the Observed World is being sent to the Crisis Center. The Crisis Center will be able to either inform the agents in the world about what's going on or it will send directives. The crisis center has to determine what is going on in the real world based on the views that the agents in the observed world have of it. It has to make a reconstruction of the facts given by the observing agents. Based on this a hypothesis of the crisis situation is formulated.

If they have a theory about the crisis, the Crisis Center can decide whether or not action should be taken and if so, what kind of action. This means that they can request Professionals to check out a current location about what is going on. If the investigations of those Professionals give enough evidence that there is a problem, they could order further units to go to the presumed location and aid in the crisis response effort. Also advice to the general public is being given about the situation.

Communication

The data flow between different components of IMACSIM can be organized in a view as shown in Figure 2. In this view we can distinguish a Simulation Layer, Agent Middleware, an agent layer and one or more GUI's. It gives a clear overview of the flow of information and in which way it is being transferred to the components in the system.

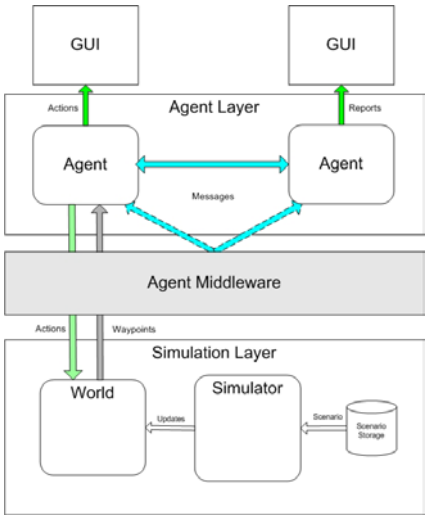

Figure 2. Global Overview Communication

The Simulation Layer contains scenario storage. In this storage scenarios are being kept. A scenario can be acquired from the storage by the Simulator. The Simulator is in charge of simulating crises and for that it needs scenarios. Those scenarios are being transformed into a script internally by the simulator. A script is basically a timeline with a start time and an end time and certain events that can take place in the world in between. The simulator is processing those events and this usually means that as a result of a certain event the world is modified in one way or another, i.e. the simulator is updating.

This event generator should be able to generate crisis events, which is an XML-based script. When the script unfolds, events are being launched and those events have their effect on the simulation environment. Every script has some open slots such as starting time, intensity, duration. To launch a script the open slots have to be filled by values of parameters. These parameters can be defined by default by a predefined script of the simulation or by the result of action during the simulation. In case of a fire the fire script can be generated. But if firemen flush the fire the intensity of the fire should be reduced.

When the world is being updated, the agents in the world should be notified via the nearby waypoints, because they have to sense changes in the world caused by the events. This is where the agent middleware comes into play. The agent middleware takes care that the agents in the simulation are receiving the updates of the world. The agents are receiving this information via agent middleware because they are supposed to be autonomous. This means that the agents are supposed to work as independently as possible. Therefore other components should not have direct access to the agents, because that would imply some sort of ownership that does not fit inside the concept of independent agents.

In the meantime, the agents are receiving data updates of the world in the form of waypoints. If these agents sense this data they can process and reason about it. Based on this reasoning the agents initiate actions. Those actions might have an effect on the world or not, but this is of course depending on the type of action that is the result of the agent's reasoning.

Besides reading waypoint data, the agents are also capable of sending messages to other agents. In the diagram the horizontal arrow indicates this, but it would be more accurate to connect the arrows via the agent middleware. This is because of the same reasons of agent independency. Those messages are being sent to other agents through the agent middleware as well. The agent actions that have an effect on the world are being propagated back again to the world to implement the changes. This requires a synchronicity scheme that ensures that the simulator applying the script to the world knows about the updates by agents, so it can update the world again according to the most recent changes.

Finally the agents are able to interact with the stream of events generated by the event generator. The users of IMACSIM will be able to view agent actions and play a certain part of an agent inside the scenario. This means that

the agents will have to have some sort of GUI because otherwise the user will not notice their actions. If they have received or sent a report, then its GUI must also be showing those, for information purposes.

SIMULATOR DESIGN

The simulator starts all parts of the system, the world, the agents and the agent communication and expert intelligence. In the simulator class we can also indicate how many agents of different available kinds we need. The basic concept of simulation is that during a fixed time interval calculations take place to update all the waypoints and actions of the agents in the world. When all those actions are processed, new information is shared between different agents and the crisis center.

In the simulator it is possible to generate different scenarios that can take place in the world. The scenarios are implemented by scripts. During unfolding of a script, several different events can take place in the course of the time. Those different events are represented by different event generators that can calculate the effects on the values of the waypoints in the world caused by that particular event. An event generator calls the necessary formulae from the physical model to receive the most recent value and replace it with earlier one.

Another reason to use different event generators is a modeling issue. Different events ask for different models and also for different updating schemes. Gas dispersion for instance is implemented using a Gaussian Plume model, that has to update all the waypoints during every time step. Because gas is very light, depending on the wind speed it moves very fast across the world. So many waypoints are involved. In case of fire only surrounding waypoints are involved and not the whole area. Unlikely waypoints are ignored as follows. Assume a fire takes place at point (x,y). In the next time step only waypoints directly surrounding (x,y) will be considered and put in a list. Then those waypoints are updated and the list is saved for the next time step. During each time step the list grows. Every times step new waypoints are being added for updating.

GAS DISPERSION

As example of the implemented models we discuss the gas dispersion model in more detail. The actual model comes down to a calculation of a concentration on a location (x,y,z) in which x,y and z are relative coordinates from the dispersion source in the sense that the x-axis is the wind direction with the origin in the dispersion source point and the cross wind direction is the y-axis, also with the origin at the dispersion source. The concentration on a time t is:

$$C(x,y,x,t)=\frac{m}{(2\pi)^{\frac{3}{2}}\sigma_x\sigma_y\sigma_z}\exp\left(-\frac{(x-u_w t)^2}{2(\sigma_x)^2}\right)\exp\left(-\frac{y^2}{2(\sigma_y)^2}\right)\left(\exp\left(-\frac{(z-h)^2}{2(\sigma_z)^2}\right)+\exp\left(-\frac{(z+h)^2}{2(\sigma_z)^2}\right)\right)$$

In this equation, m is the released mass, uw is the wind speed and h is the source height. σ x and σy and σz, the dispersion parameters, depend on the wind speed and the time of the year and other climatologically parameters such as air stability and wind speed.

The parameter mass m, offers the opportunity for user interaction. In case the gas opening is gradually covered the amount of gas should be reduced and as a consequence the amount of distributed gas in the area. In case agents start to cover the gas opening implies sending a message to the event generator with an update of the mass parameter.

Our dispersion model is an approximation of the reality. At every time step we update all the waypoints at once, including the amount of gas dispersed before which was already released in the air. A incremental approach is more natural but computationally very complex.

IMPLEMENTATION

With JADE being one of the leading platforms for multi-agent programming, it was decided that the simulation should run on JADE, to facilitate running of system on mobile devices in the future (Bellifemine, 2001).

In Figure 3 IMACSIM is being described as a series of Java software components that are independent parts of the system and exchange information during the course of a simulation. In some of these components external open source software components like JADE and Jess were used. In the description of each component the use of that particular tool will be described.

Simulator: This is the main component of the program, where the simulation is being set up and run. The startup and all necessary parameter checking are done here. From here all the other necessary components are started. The simulator ensures that the agents receive the waypoints that the agents need to correctly observe the world. By passing the essential data via the agent middleware ensures the agents in the field receive the appropriate data.

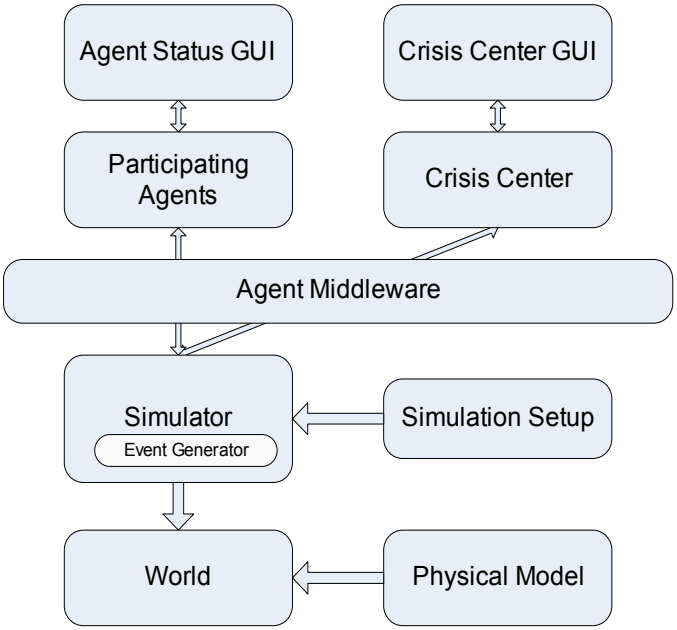

Figure 3. Software Component Decomposition

World: The simplification of the world for which we are making simulations. We cannot incorporate everything that is in the real world into our world model. Because of complexity and performance reasons we make the world in which a scenario is developing as simple as possible. The World consists of waypoints which contain the values that the agents observe.

Physical Model: The collection of models that enable the events to impose changes inside the world. These are mainly a set of formulae that can be changed or added when necessary. The formulae are designed in such a way that they can be replaced by more advanced counterparts very easily.

Event Generator: This component makes sure that events, that are part of a scenario, are executed in the right way. This means that the event generator takes care of updating the world model in the fashion that a particular event requires. It uses the formulae available in the Physical Model to update a particular event. There are as many Event Generators as there are active events in the simulator. Whenever according to the scenario a new event needs to be launched, a new event generator is started that takes care of updating the waypoints with data relevant to that event.

Simulation Setup: In this window all the properties of a certain simulation can be edited, loaded and saved. After the user is satisfied with the scenario, it can be stored and executed. Things like starting time, ending time, number of participating agents can be set up in this GUI. It is really up to the creativity of the programmer and the complexity of the simulation presented how much parameters and details can be set.

Agent Middleware: The agent middleware ensures that the messages that are sent between agents are being processed correctly. Also the updates of the world are processed so that the agents have them at their disposal. IMACSIM uses JADE as Agent middleware, which is a very popular platform for the programming of multi-agent systems.

Participating Agents: These are the key elements in the system. They are the main characters in the simulation and as the scenario unfolds, the characters respond to the changing situation. The agents can communicate with each other about the current status, and they can use reasoning for making decisions about what they should do the next. Participating agents can also contact the crisis center. For communicating, they use JADE agent messaging, through FIPA compliant ACL Messages (Bellifemine, 2001). These messages contain information that is based on a Java-Based crisis ontology developed by Fitrianie (Fitrianie, 2007). For reasoning they use a knowledge base. As a knowledge base Jess is being used for its easy interaction with Java objects, such as the ontology elements inside Fitrianie's crisis ontology.

Crisis Center: during a crisis, a crisis center is usually being set up to coordinate the crisis response effort. Here all the information about the crisis comes in and is being interpreted by the people who are there. The crisis center interprets all the messages that are coming in and based on rule-based reasoning, they create orders for their subjects that can be sent by using agent messages. They issue commands to the participating agents that are in the field, based on the information that they get from all available sources. This information is being stored inside something that is called an interpreted world. This is an interpretation of what the crisis center believes is going on inside the world. In real life, the interpreted world is usually some sort of real or digital map onto which status information is being added. Based on what they see on the map, the crisis team decides what to do. The decision making functionality described in Figure 3 is implemented in this component.

Interface during simulation (Agent Status GUI and Crisis Center GUI): During the simulation the agents are making all kinds of decisions, and a lot of communication is exchanged as well. We want to keep track during run-time of what is going on in the world, so we need some way to eavesdrop on the communication during a crisis. In quite the same way that communication during a flight is being recorded, the agent communication will be shown on a user interface while the simulation is running. Besides that, for the agents that the user identifies with during the simulation, agents can be controlled and agent actions can be initiated from this GUI. For the participating agents, the status of the agent is being shown. As a visual component of the crisis center, an icon-based crisis map will be used, that is able to show incoming crisis messages on top of a map of a crisis area. This was made using components from another application called 3MNews, designed by (Tatomir, 2006).

EVALUATION

For testing MACSIM, simulations had to be run, and for simulations we needed scenarios. These scenarios are script-based. This means that a scenario consists of an XML-based script. A script is a combination of a time and an event with the source location of the event added to it. In this way the event generator knows exactly what to do to update the world based on that event. The first minutes of an example test scenario are displayed in Table 1.

Our software was tested at the campus of Delft University of Technology. In our scenario the University was threaded by a toxic cloud. A group of 21 students Technical Informatics was located at different places on the campus. On their mobile wireless connected laptops the Gui from Figure 4 was displayed. The first window displays an overview of the campus. The two windows below the main window are reserved for incoming and outgoing messages. The other windows show the status of the agents and are visible for the controller. Every student had a part of a script and was supposed to report about events using an icon language (Fitrianie, 2007) or text. They report about events as Big Bang, Lightning, and smell and location of victims in the street. All messages were sent to the Crisis Center for processing and interpretation.

It proves that our event generator works as expected. Observers were able to sense the world via information sensed from the nearby waypoints visible in a window at the screen of their laptop. They were able to report about the events via the icon based language tool. All messages were sent to the crisis centre and visible at the screen. The simulator controller has access to all information and is able to inspect the displayed information. It can be concluded that the communication layer works fine. The only problem was the unpredictable delay caused by the wireless network, so that events were reported without logical order.

CONCLUSION

With the development of IMACSIM we have devised a system that is able to simulate crisis situations based on scripts. For this purpose a crisis generator component, a communication layer and intelligent agents were developed and tested. The agents in the system are autonomous or under direct control of the user via a graphical interface. Actions of the user have their impact on the event generation so IMACSIM is a system for interactive simulation.

An advantage of this system is that it enables us to add new physical models, new intelligent agents and new types of crisis events very easily; also existing components can be updated very fast.

As an open simulation platform, IMACSIM offers a lot of room for further development also in the area of serious gaming (Benjamins, 2006). The first additions planned for the near future are the use of gaming software for the creation of graphical components for 3D graphics and more sophisticated algorithms for agent reasoning, like Bayesian belief networks. The existing user interface will be upgraded to improve the visual experience and sense of presence for the user that runs the simulation. At this point the user interface just shows message traffic, but real time crisis simulation of course is something that cannot do without a realistic visual representation.

Furthermore, the design offers enables adding more advanced reasoning algorithms. In all cases where new intelligence/ knowledge has to be added or intelligence/ knowledge should be updated, it is important to have discussions with the domain experts during the development process to get useful first-hand experience and information that can be represented in rules, behaviors and agent actions. The rules in this way will also become better and more sophisticated.

Time	**Event + Location**	**Agent**	**Possible Action**
14:00	Fire starts at (X,Y)	General Public	Run Away + Report trough GSM or SMS
		Sensor	Activation smoke alarm + reporting to Crisis Center
			Sends a call to Fireman about developing situation.
		Crisis Center	Receives a call from Crisis Center to go to (X,Y) and is on his way
14:02	Fire develops	General Public	Complaint to Crisis Center about developing smoke
			While on its way to (X,Y) receives info from crisis center that fire is developing
		Professional (Fireman)	
14:04	Explosion	General Public	Report to Crisis Center about damage and casualties and a loud bang
		Crisis Center	Order extra units of Firemen and Police to (X,Y)
14:05		Professional (Fireman)	Arrives at (X,Y) and starts extinguishing fire
		Professional (Policeman)	Receives order from Crisis Center to seal off crisis area around (X,Y)

Table 1. First minutes of Example Scenario

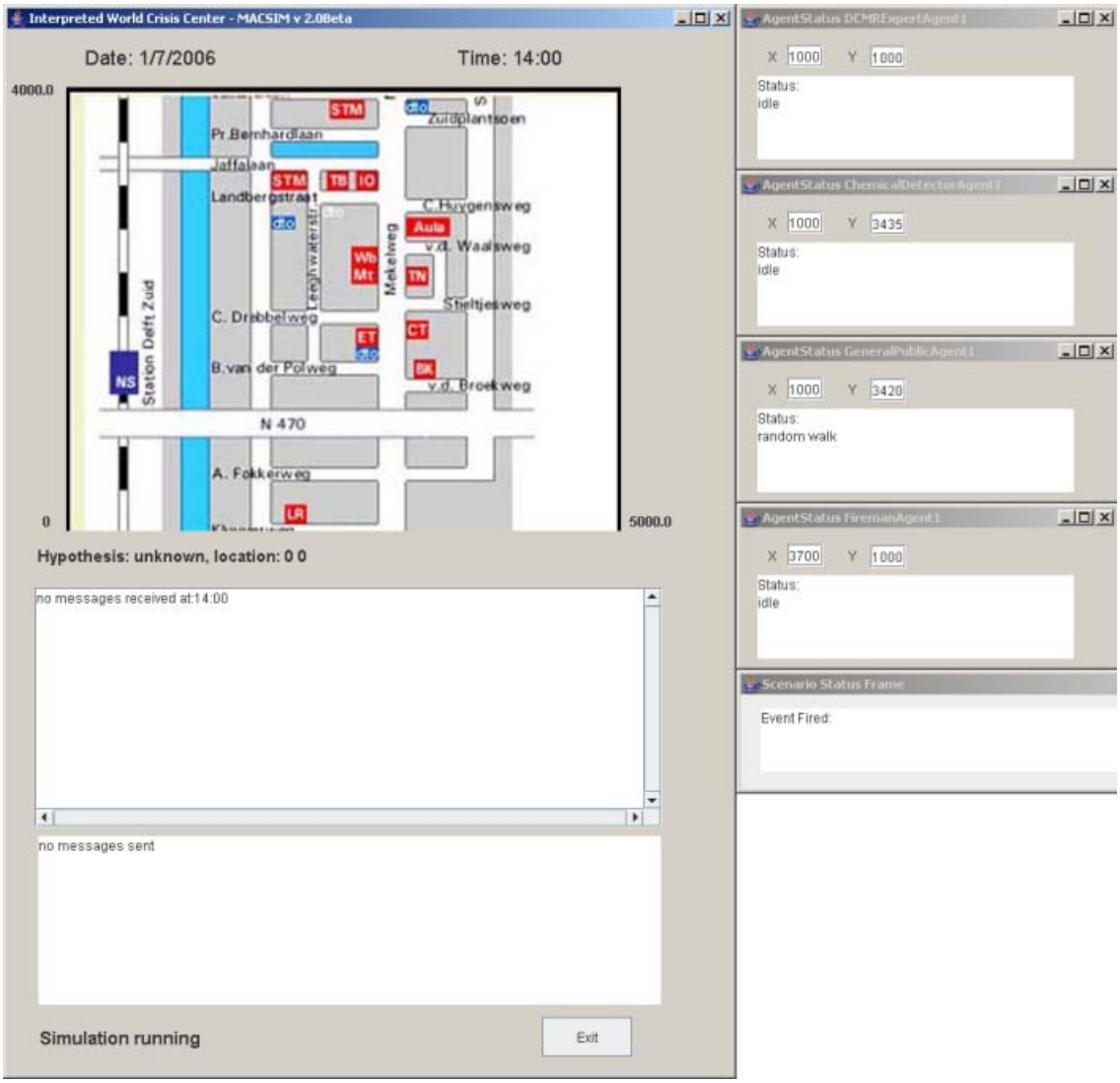

Figure 4. Overview MAXIM GUI component

REFERENCES

1. Bellifemine, F., Poggi, A., Rimassa, G., Jade: a fipa2000 compliant agent development environment. pages 216–217, 2001.
2. Benjamins, T., Rothkrantz, L.J.M., MACSIM: Serious gaming in crisis management via script-based simulation, CGAMES 2006, October 2006.
3. COMBINED Project Website, (http://combined.decis.nl).
4. EPA and NOAA. ALOHA User's Manual, 2004.
5. Ernest Friedman-Hill. Jess in Action. Manning, 2003.
6. Fitrianie, S., Rothkrantz, L.J.M., A visual communication language for crisis management, International Journal of intelligent Control and Systems, to appear 2007.
7. Hazmat: Hotzone project website, (http://projecthazmat.org).
8. Jain, S., McLean C.R., Integrated simulation and gaming architecture for incident management training, Proceedings of the 37th conference on Winter simulation, ISBN:0-7803-9519-0, 904-913, 2005.
9. Klapwijk, P., Rothkrantz, L.J.M., Topology Based Infrastructure for crisis situations, Third International Conference on Information Systems for Crisis Response and Management ISCRAM 2006, May 2006.
10. Loper, M.L, Presnell, B., Modelling an emergency operations center with agents, Proceedings of the 37th conference on Winter simulation, ISBN:0-7803-9519-0, 2005.
11. Mehrotra, S., Butts, C., Kalashnikov, D., Venkatasubramanian, N., Altintas, K., Haimi, L., Wickramasuriya, J., Hariharan, R., Ma, Y.: CAMAS: A Citizen Awareness System for Crisis Mitigation. University of California, Irvine.
12. Robinson, C.D., Brown, D.E., First responder information flow simulation: a tool for technology assessment, Proceedings of the 37th conference on Winter simulation, ISBN:0-7803-9519-0, 919-925, 2005.
13. Rothkrantz, L.J.M., Fitrianie, S., Constructing knowledge of the world in crisis situations using visual language, IEEE International Conference on Systems, Man and Cybernetics, October 2006.
14. SAFER Systems Inc, SAFER REALTIME/TRACE Technical Manual.
15. Simulator project website, (http://www.ics.uci.edu/~projects/drillsim/).
16. Tatomir, I., 3MNews - Message-based Multimodal News, Technical report KBS 2006-02, TU Delft, 2006.
17. Tatomir, B., Rothkrantz, L.J.M., Popa, M., Intelligent system for exploring dynamic crisis environments, Third International Conference on Information Systems for Crisis Response and Management ISCRAM 2006, May 2006.