A Multi-agent System for Dynamic Path Planning

Marcelo Cardoso Silva[1], Ana Cristina Bicharra Garcia[1] and Aura Conci[1],

[1] Universidade Federal Fluminense, Instituto de Computação, Rua Passo da Pátria. 156 Bl.E, 24210-240 São Domingos – Niterói – RJ, Brasil
{mcardoso, bicharra, aconci}@ic.uff.br

Abstract. Escape route planning in emergency situations generates interest among researches in many different areas. Computer Science contributes to this by developing simulations of real world situations. An escape route can be regarded as a weighted graph where the labels associated with its edges are determined by the distance between two connected vertexes. Hence, given a starting vertex, it is possible to obtain the minimum path to one exit in the environment. Classical algorithms, such as Dijkstra's, can solve this problem. However, when the environment changes in real time, this class of algorithms is not efficient. Using Multi-agent Systems to solve this problem dynamically seems to be a good approach to plan escape routes in real time, given new information about the environment when modification are detected. This research exposes this approach, implementing a multi agent communication system in a 3D virtual environment.

Keywords: Multi-agent Systems, Route Planning, Dijkstra's Algorithm, FIPA-OS, VRML.

1 Introduction

There are many applications to the problem of discovery of the best way to achieve an exit in an environment. This optimum path often is known as escape route. This problem is treated in a generic way through a weighted graph which edge's weights are related by the distance between points, connected by them. This approach considers the best path, those whose cost is the minimal available form two points: the origin and the destination.

The path planning can be made through classic algorithms as the Dijkstra's algorithm, which main objective is to find the minimum way from an initial point in a graph to all other points of this graph [7]. The discovery of the optimum or the minimum path to reach an exit in an environment (escape route), can be reached using this algorithm. However, in a real situation, the search space is not always previously known, at least in its totality, and many factors can contribute for modification of this environment in real time. These aspects makes the modeling of a simulation, a task sufficiently complex and also, in some cases, impracticable. This shows the necessity of combining different methods in the effort to solve dynamic problems.

This work considers Multi-agents [10] for the dynamic path planning in real time, based in message exchanging among agents. Combining this approach with the Dijsktra's algorithm [7] seems as an alternative solution for the problem. To achieve this a simulator to visualize the virtual 3D environment was created. In this, it is possible to observe an agent in movement to search the environment exit. For this, the agent plans a route, that is established dynamically considering information acquired from messages given by other agents. It is through this exchanged messages that each agent knows the environment, that is, discover a series of environment characteristics. The virtual environment can suffer alterations in real time through interventions of a human operator, as well. On clicking at one or more position demarcated in the environment, the human operator invalidates this position or promotes an interruption in a point of the graph. These make the agents to search new exits in the environment.

Figure 1 presents the 3D simulation environment. The colored pins represent the agents, and the colored lines represent the calculated path since the beginning. The white squares are points that the user can turn into obstacles. Red points are blockages, for the moment. On the right site of Figure 1, the blue agent tries to move north when it finds that the path is blocked. Then the agent returns to the previous node to make a new decision. This characterizes the dynamic path planning the system achieves.

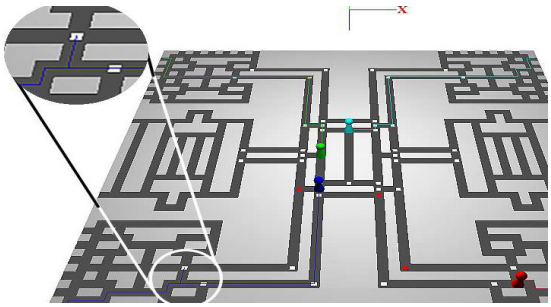

Fig. 1. Virtual environment screen: pins represent the agents, lines show agent path, white squares are the changeable nodes and red points are closed nodes at the moment. The spot shows the blue agent diversion, which characterizes the dynamic planning achieved by the system.

This work presents a summary of issues and results discussed in [9] and is organized as follows. Section 2 presents related works. Section 3 provides details on the framework our system uses, FIPA-OS. Section 4 discusses the tests and experiments. Finally, Section 5 presents the conclusions and future works.

2 Related Work

Braga [1] presents a study that uses concepts of Virtual Reality and Multi-agents to define specification for the development of a Multi-agent environment to simulate escape routes in cases of accident. The use of the Virtual Reality aims the immersion

in the environment, whereas the use of Multi-agents aims to simulate real world. This work considers security, signals and human behavior in cases of accident. A prototype of the simulation in a oil platform on fire is presented, in order to show the behavior of the agents during an evacuation. Adequate signalization of the environment is the main idea of Braga's work [1]. Through this signalization the agents decide about the actions they will take in the environment. Such simulator can be adapted for different environments, agents and types of accidents.

Caetano and Pereira [2] presents a model to agents' representation that allows simple and flexible management. They use an efficient mechanisms of interaction between the avatar (representing the user in the virtual environment) and the agent, as well as among the agents. A comparison among different animation models in VRML is presented, as well. Three models are considered: explicit animations in VRML, VRML animations using Script Authoring Interface (SAI) and VRML animations using External Authoring Interface (EAI). The first model defines the behavior of the agent in the VRML code through the Script node, producing high complexity scenes. In the second, the separation between geometrical modeling's and behavior, through the use of Java code is done. This allows the creation of virtual worlds with complex interaction stiles and better performance. The third model was specified based on the EAI use, allowing the total separation of agent behavior and graphic representation. A comparative study of these three models considers the performance on execution of each models bases on the quantity of events related in the communications agent-agent and agent-geometry. The third model presents the best architecture, because it permits the separation of geometry and behavior, which is a simple and efficient representation for heterogeneous multiple agents systems. Such work was developed in a proprietary Java code.

3 The System

The main objective of the system is observe agent behavior in real-time while they search for the nearest environment exit. They look for invalidated positions in the environment and inform Dijkstra to not consider them anymore. Dijkstra calculates the nearest exit based on the agent position and the exit information located in Memory. Dijkstra then informs the agent the shortest path. The agent advances to the next node and starts the process again.

Figure 2 illustrates the general system architecture. The implemented architecture is very generic regarding the use of the graphic and cognitive layers, being able to be applied in different models of applications based on Multi-agents Systems and Virtual Reality Modeling Language (VRML).

The system uses a client/server approach where the client side of the application works in a desktop web browser. It accesses the server part of the application through a RMI (Remote method invocation) layer [8].

The server part of the application uses FIPA-OS [5], which is a framework responsible for managing the agents' cognition. The transport RMI layer is the main aspect of integration between the virtual environment and FIPA-OS. RMI is a CORBA (*Common Object Request Broker Architecture*) version for Java. This means

that with RMI an object can call a method of other object placed everywhere as a local method.

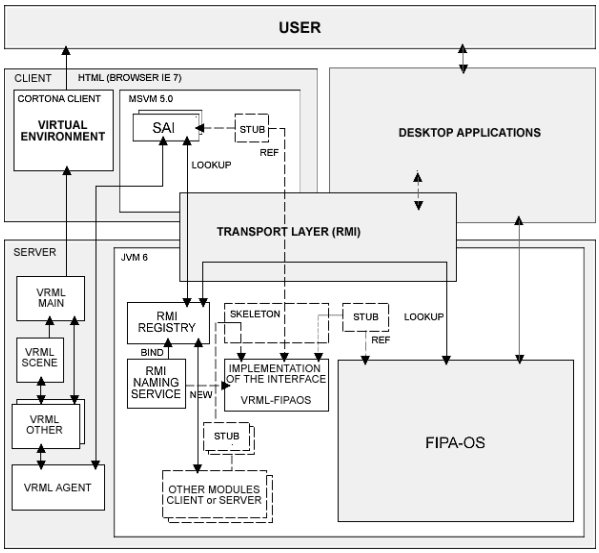

Fig. 2. VRML and FIPA-OS integration.

The use of a remote method cannot directly be made. The client side object makes a call to a local routine named *stub*. *Stub* is registered in the RMI server by a search of names in the lookup table. After that, it opens an outlet, serializing the objects and passes the data for another routine known as *sketeton* that it is in the serving machine. *Sketeton* represents the real implementation that occurs in the server. It is responsible for receiving the data come from the client and repasse them to server and to return them to stub [8].

The agents live in a 3D maze environment which is modeled as a graph. Figure 3 shows the agent behavior model of interaction with the environment and the FIPA-OS framework. The role of each element in this model is described on the following.

Human operator represents the simulator user. It can modify the environment during a simulation, including or deleting graph edges and nodes, giving dynamic aspect to the virtual environment.

Environment represents the model of the virtual environment. In this environment, the movement of the virtual agents is represents as well. These paths are represented by colored lines. The color used to represent the path is the same used to represent the agent in the scene, as can be seen in Figure 1.

Agent (frame) represents the agent in the environment. It concentrates the intelligence of the simulation on route planning.

Path restrictions store the modifications made in the data structure of the virtual environment. These modifications are made by the human operator during a simulation.

Memory stores all XYZ positions of possible agent exits in the environment.

Cognition is the module responsible for decision. It selects the smallest path among the possible paths found by Dijkstra algorithm.

FIPA-OS and VRML communication are modules responsible for receive and send the message agent. **FIPA-OS** uses the agent communication layer of FIPA: the *message transport protocol* [4] to perform its task.

Dijkstra, represents the algorithm used to find the shortest route from a point to the destination

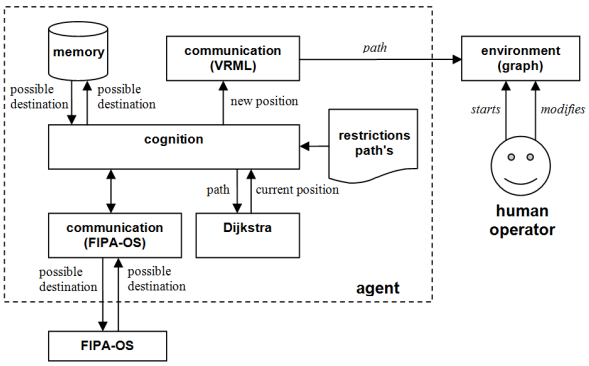

Fig. 3. Agent model and its integration with VRML and FIPA-OS.

The interaction among modules begins in the moment that the human operator starts the simulation, that is when the agents and virtual environment are created. The simulator controls following actions:

Simulator start: the human operator loads the VRML file in his browser. This file contains the virtual agents arbitrarily placed in a XYZ point of the virtual environment. The requirements for each agent are loaded and an instance of the agent is created in the framework FIPA-OS.

Environment changes: the human operator has the option to invalidate some positions (edges and nodes in graph) marked in the virtual environment through a mouse click. These positions are represented in the scene by small white squares. The white color indicates that the node is connected to the graph, allowing that the agent go through it. On receiving a click, these squares change their state, turning to red color. Red indicates that the node is disconnected and that the agent should do not considers it in their path.

Sent message: from the moment that all players have been created the process of communication between them by sending messages begins. The message sent by the agent contains his position on the environment, i.e. the position he occupies at the start of simulation. This position XYZ is characterized as a possible exit of the environment for other agents. Messages are sent repeatedly, until the agent find an exit on the environment. Once he finds the exit, he stops to communicate with other agents, ending his participation in the simulation.

Received message: when an agent receives a message, the agent can begin his movement in the scene, i.e. receiving data from Dijkstra module. The content of the message is written in a list named the agent memory. This list contains the position of all possible environmental exits for each agent. Although the exchange of messages

between agents occurs repeatedly during the simulation, allowing the sent of same message more than once for the same agent, the agent memories consider only one instance for each informed exit.

Agent cognition: the main objective of the simulation is visualize agents movements in virtual environment on search of the shortest route to the possible exit of the environment. The decision about the path to be taken by the agent is based on the response of a query made to the Dijkstra. To do this, the agent checks the positions of the environment that has been invalidated by the human operator and advises the Dijkstra to not consider this in its calculations. Then, its informs the current XYZ position (source), together with each one of the possible exits stored in the memory. The Dijkstra calculates and returns to the agent, the shortest route. In the path, the agent uses only the next node and moves to it. All process is then considered again.

The model proposed in this work does not use an Agent Coordinator. All agents know their tasks, exchanging messages with all others, providing and using the necessary information to achieve their individual goals. This behavior is the default federative pattern of FIPA-OS [5].

4 Experiments

The experiments were performed with a notebook using a 600 MHz Pentium III processor, 320Mb of RAM memory, and an ATI Radon Mobility graphics board. In the simulations, the quantity of agents was increased, up to the limit supported by the equipment used in the experiments. Twelve experiments were carried out.

Table 1. A summary of the results.

Exper.	Time	Agents	Ext.	Centr.	Out	In	Re-plan.
1	1	4	4	0	3	1	0
2	2	4	4	0	4	0	1
3	3	4	4	0	4	0	2
4	4	4	4	0	4	0	2
5	2	6	4	2	6	0	5
6	2	8	4	4	8	0	4
7	2	8	8	0	8	0	5
8	2	12	8	4	12	0	11
9	4	12	8	4	12	0	9
10	8	12	8	4	12	0	6
11	2	16	16	0	16	0	13
12	2	20	16	4	16	4	15

The environment was divided in six regions. The external regions are called: NW – Northwest, NE - Northeast, SE - South-east and SW – South-west. The central regions are called CW – central west and CE – central east.

Table 1 illustrates the results. About the columns: “Exper.” is the experiment number; “Time” is the interval among each agent step on the route; “Agents” is the number of agents in the simulation; “Ext.” is the number of exits on external regions:

NW, NE, SE and SW; “Centr.” is the number of exits on central regions; “Out” is the number of agents that achieve the exit; “In” is the number of agents that do not achieve the exit; “Re-plan” is the number of agents that changed paths. This Table shows the importance of communication among agents on the route planning. It suggests a great relation among the agents position on the regions of the environment and performance.

Figure 4 considers relations on the re-planning of routes. Blue lines with diamonds symbols represent the number of agents, magenta lines with square symbols represent the number of agents that reach the exit, and yellow lines with triangular symbols represent agents that re-plan routes.

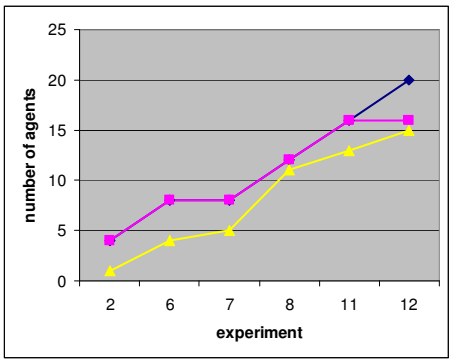

Fig. 4. Results on re-planning the routes.

This figure illustrates the results of a comparative study on the agent position on the environment without the central region but considering the number of path re-planning. It shows that when there is a small number of agents to achieve the exit, agents need to first go to other regions.

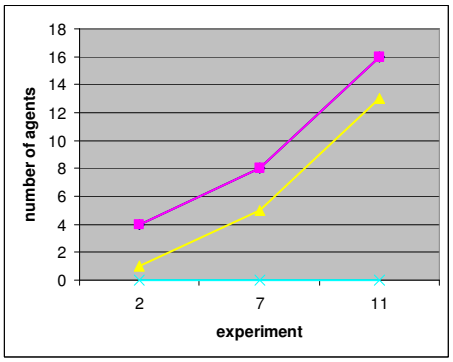

Fig. 5. Results on influence of exits of central regions, without agents.

In Figure 5, magenta lines with square symbols represent the number of agent achieving the environment exit (Column 6 in Table 1). Blue lines with diamond symbols (superimposed with magenta lines) represent the exits on external regions

(Column 4 in Table 1). Yellow lines with triangle symbols represent the agents that re-plan the routes (Column 8 in Table 1). Cyan lines with "x" symbols represent the number of exits located on central regions (Column 5 in Table 1).

This Figure considers the number of path re-plannings, along with agents that are not located in the central regions when the simulation begins. It illustrates that when there is a small number of agents that achieve an exit, they need to go first to other regions.

Figure 6 illustrates that the number of re-plannings decreases when the agents are placed in central regions. In this scenario, the greater proximity among agents improves the communication quite a bit, resulting in more intermediate exits.

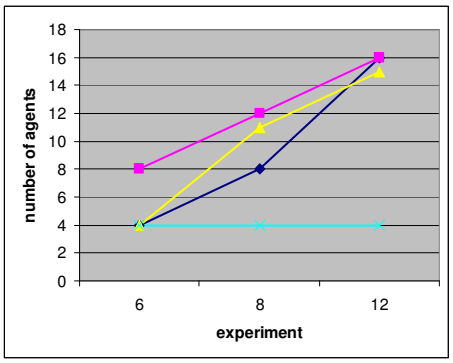

Fig. 6. Influence on position agents in the central regions.

In Figure 6, blue lines with diamonds symbols represent the exits on external regions (Column 4 in Table 1). Magenta lines with square symbols represent the number of agent achieving the environment exit (Column 6 in Table 1). Yellow lines with triangular symbols represent the number of agents that re-plan the routes (Column 8 in Table 1). Cyan lines with "x" symbols represent the number of exits located on central regions (Column 5 in Table 1).

5 Conclusion

The main contributions of this work are as follows. Using a message exchanging paradigm for agents offers satisfactory results for decision making in environments that change in real-time.

Another contribution is proposing an integrated architecture with VRML and FIPA-OS, which allows the user to visualize a 3D environment of the simulation, making it easier to analyze the agent behaviors in real-time.

The results of the experiments show that using multi agents in the dynamic path planning helps the individual agent in the decision process for an escape route. From the information obtained in real-time and through agent communication, it is possible to information about the environment and better uses resources.

Different factors influence agent decisions. An important issue for future works is studying the reputation of the agents. The reputation of an agent influences straightly the behavior of others agents in the system, because it can provide false information to other agents to take advantage from a situation, depending on its goals. In this work, all the messages exchanged among the agents provide accurate information. Future works on this research must consider these factors when modeling agent behavior, to make them act more similar to real world situations.

References

1. Braga, L.A.F.: Simulação de Rota de Fuga e Sinalização Utilizando Multi-Agentes e Realidade Virtual. D.Sc. thesis, Universidade Federal do Rio de Janeiro (2006)
2. Caetano, A., Pereira, J.: Representação de Agentes Autónomos em VRML 97". VIRtual-Revista Electrónica de Visualização, Sistemas Interactivos e Reconhecimentos de Padrões (2000)
3. Finlayson, R.A.: VRML 2.0 EAI FAQ [online] The VRML EAI FAQ, http://www.frontiernet.net/ ~imaging/eaifaq.html
4. Foundation for Intelligent Physical Agents, http://www.fipa.org
5. FIPA-OS Developers Guide [online] Emorphia, http://fipaos.sourceforge.net/docs/DevelopersGuide.pdf
6. Gluz, J.C., Viccari, R. M.: Linguagens de Comunicação entre Agentes: Fundamentos, Padrões e Perspectivas. In: Anido, R.O., Masiero, P.C. (org.) Jornada de Mini-Cursos de Inteligência Artificial. vol.VIII, pp. 53 -- 102, Campinas (2003)
7. Goodrich, T., Tamassia, R.: Projeto de Algoritmos: fundamentos, análise e exemplos da Internet. Capítulo 7, pp. 342 -- 379, Bookman, Porto Alegre (2004)
8. Linden, P.v.d.: Just Java. Capítulo 9, pp. 352 -- 359, Makron Books, São Paulo (1997)
9. Silva, M.C.: Um Sistema Multiagente para o Planejamento Dinâmico de Caminhos. M.Sc. thesis. Universidade Federal Fluminense (2007)
10. Stone, P., Veloso, M.: Multiagent Systems: A Survey from a Machine Learning Perspective Autonomous Robots, 8(3):345 -- 383 (2000)
11. ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004 Virtual Reality Modeling Language (VRML) [online] Web 3D Consortium, http://www.web3d.org/x3d/specifications
12. Wooldridge, M., Jennings, N.: Intelligent Agents: Theory and Practice (1995)