Military Technical College
Kobry El-kobbah,
Cairo, Egypt

5th International Conference
on Electrical Engineering
ICEENG 2006

MULTI-AGENTS MODEL TO ENHANCE THE NAVIGATION PROCESS IN A VIRTUAL ENVIRONMENT

Ismail Abd-Elghafar Khaled El-Menshawy Ali Ali Fahmy

ABSTRACT

Due to the rapid evolution of graphics hardware, interactive Virtual Environment is becoming popular on desktop personal computers. The use of the Virtual Environment as a simulation system becomes very important for certain types of applications, especially in the fields of education and entertainment. These synthetic environments are even more attractive for the user when they exhibit dynamic characteristics. The most important problem of using the synthetic Environment is navigation process. The ability to navigate and interact in a Virtual Environment is essential for certain types of applications, such as virtual classrooms, on-line museums and games. Many 3D *virtual environments*, whether representing existing places or imaginary ones typically leave the user alone and partially or totally unassisted in navigating the environment *Navigation* process deals with the problem of finding path/trajectory between two locations under some constraints. In this paper, a model of multi-agents is developed to enhance the navigation process and interaction with the users of the synthetic Virtual Environment It describes how agents can work together to solve their task. Usually, this requires some kind of inter-agent communication.

KEYWORDS Virtual Environment, Multi agents, navigation, Virtual Reality Modeling Language (VRML), avatar**.**

1. Introduction

The ability to navigate and interact in a Virtual Environment (***VE***) is essential for certain types of applications, such as virtual classrooms, on-line museums and games. Many 3D *virtual environments*, whether representing existing places (e.g., virtual cities) or imaginary ones typically leave the user alone and partially or totally unassisted in navigating the environment. One of the urgent problems of agent's system is to build agents that are capable of autonomous action, accept and execute high-level task descriptions with no human supervision. On the other hand, the area of ***VE*** seems to be appropriate for the development of intelligent agent's applications. Therefore, it seems that the time has come to integrate the ***VE*** system with Intelligent Agents. This combination of intelligent techniques and simulation tools together with effective means for their graphical representation and interaction of various kinds, have given rise to a new area at their meeting point, which we call Intelligent Virtual Environment. A virtual agent can be defined as an autonomous entity in a *VE*. It should not only look like, but also behave as a living organism, and be able to interact with the environment and its inhabitants. Developing guided tours led by embodied agents is not an easy task for the 3D content creator, since it currently has to be done partly by hand (e.g., coding a suitable path for the virtual agent avoiding obstacles). Moreover, the code written for one 3D VE can be very limitedly reused for other ones. In [5] One of the effective ways in developing navigation systems and exploring the ***VE*** is using the intelligent agent because of its properties, which increase its efficiency and easiness of exploring the ***VE***. It becomes

increasing important to give the autonomous agent the ability to navigate around dynamic objects in the Virtual Environment. This paper is motivated by the necessity to create a generic model to improve the navigation process in the **VE** by applying the embodied agents model. This model must be:
Applicable to very complex 3D Virtual Environments;
Computationally inexpensive to operate in real time;
Able to find a path, if it exists, and the path length should not differ much from the length of optimal path.
Improve the capability of the ***VE*** browser by adding an autonomous agent to enhance the support of the user by doing navigation process.
Agents in the proposed model contain knowledge, are designed to work autonomously, act on behalf of the user, and have the ability to learn. The virtual intelligent agents are navigation aids, leading users around and preventing them from being lost. The users have simply to follow the guide or trace the drawn path to explore the ***VE***. Also, it works as information aids (since the agent is also able to provide information about the encountered places and objects). An embodied agent can also have the additional advantage of making the 3D ***VE*** more lively and attractive for the user. The core idea of the proposed model is based on the accessibility knowledge representation by building a navigational map and the use this map for real time navigation process. The approach we follow to deal with the problem of deriving a suitable set of positions for the guided tour is based on using a motion planning algorithm that takes as input a map of the virtual environment. The rest of this paper is organized as follows is as follows: ***Section 2*** a brief overview of related works of navigation process in the Virtual Environment. ***Section 3*** reviews the conceptual model of the Virtual Environment Browser.. ***Section 4*** explains the general proposed model of an intelligent Virtual Environment, discusses the rules of building the Virtual Environment based on multi-agent system, illustrates the roles and behaviors of each agent described in the proposed model. ***Section 5*** shows the experimental result to measure the performance of the ***VE*** browser after the multi-agents model has applied The paper is ended with the conclusion and the future work.

2. Related Works

There is an enormous body of work related to 3D navigation in virtual environments. It can be classified into two main categories: working on understanding the cognitive principles behind navigation, and working on developing navigation techniques for specific tasks and applications. A task-based taxonomy of different navigation techniques is presented in [*4*]. The author in [*2*] have explored cognitive and design principles as they are applied to large virtual environments. There has also been considerable work in designing intelligent user interfaces for improved navigation. Most of the prior work on navigation for walkthrough applications has focused on developing body-centered interaction methods, including devices such as treadmills, or on evaluating the differences between various interaction techniques, such as walking and joystick based flying, There has been less work on automatic computation of navigation paths in complex environments. [**8**]The author has presented an inter-action technique based on body gestures for walking and ascending, or descending, steps and ladders in virtual environments. These techniques were applied to architectural walkthrough environments. The author has presented a simple interaction technique for walkthroughs in which the user draws the intended path directly on the scene, and the avatar automatically moves along the path. Many of the current computer games also offer effective means of navigation. However, it is not clear that these approaches can be extended to navigation and automatic path computation in general massive environments. For handling very large environments, Wilson et al. have presented fast algorithms for collision detection

and distance computation between the avatar, or other moving objects, and the rest of the environment. The resulting system works well in terms of computing collision-free paths in the localized neighborhood of the avatar, but cannot be used to compute a global path between arbitrary initial and destination positions in a complex environment. [***10***]
A model is presented for describing the Intelligent Information in the Virtual Environment, as well as the Camera Control. Its main goal is to be able to populate Virtual Environments with virtual agents endowed with different Levels of Autonomy: from guided to autonomous. [3]
The authors in [7]review the intersection of AI and VE. They consider the use of AI as a component of a VE and Intelligent Virtual Agents as a major application area, covering movement, sensing, behavior and control architectures. In their systems, intelligence is embedded in the system architecture itself, by incorporating AI algorithms into the virtual reality system. They survey work on emotion and natural language interaction, and considers interactive narrative as a case-study. Another proposed path–planning algorithm is proposed to modify a Java3D implementation of VRML (Virtual Reality Modeling Language) browser to incorporate it into the user interface in the control loop of 3D interactions to compute collision-free maneuver paths but without using Agent based system.[9] Now, we have to answer the following question, how can we browse and interact with the desktop Virtual Environment? The answer is that the user has to use a Virtual Environment Browser to be able to see the contents of the VE and interact with the objects in it. The next section explains the conceptual model of the Virtual Environment Browser. To realize our model we used VRML, which limits us to a medium quality desktop VR.

3. Functional model of the Virtual Environment Browser

We have mainly modified the routine of processing mouse events, the routine of updating the next viewpoint configuration and the routine of Navigation Process in the virtual environment browser. Fig.1. illustrates the Interaction between the end user and the synthetic *VE*. This environment is created by using *VRML*.

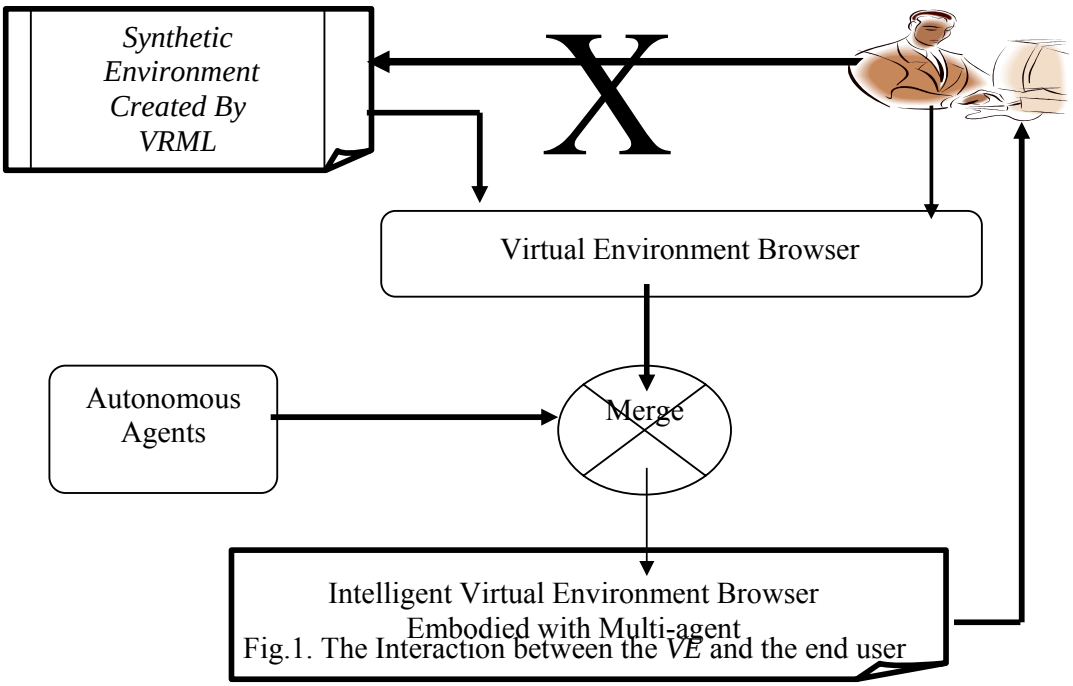


Fig.1. The Interaction between the *VE* and the end user

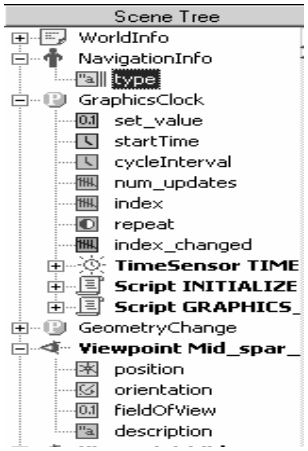


Fig.2. Description of the VE file as a scene graph

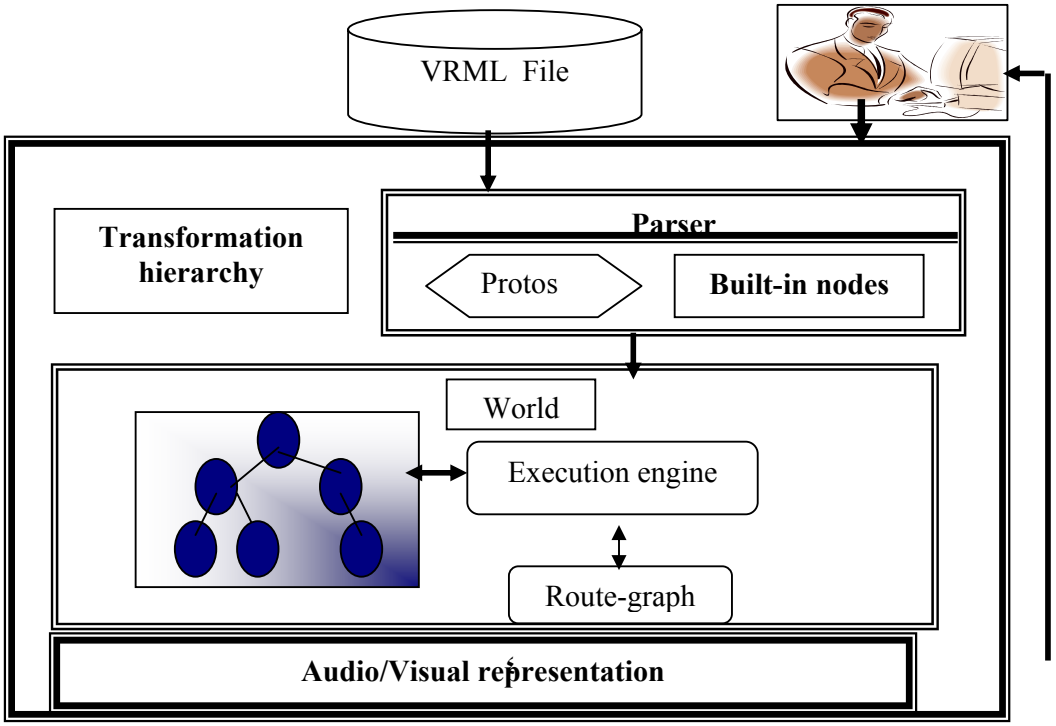
VRML file describes a 3D environment based on a scene graph structure. Scene graphs are hierarchical treelike data structures that describe an entire 3D scene, including the geometry representation, the appearance of objects, and the relationships among the objects in the world as shown in figure (2)

Many 3D *VE* Browsers do not offer sufficient assistance to users in navigating through the *VE*, find objects/places of interests, and learn how to interact with them. This paper proposes the adoption of Intelligent Navigation agent of VE as an effective user aid and describes a novel tool that provides automatic code generation for adding such intelligent guided agent to 3D *VE* developed using the *VRML* and *JAVA*. The VE Browser interprets a *VRML* file and presents the corresponding 3D ***VE***. Figure 3 illustrates the conceptual model of the Virtual Environment Browser. There are three steps to display a *VRML* file in the Browser.

(a) The parser interprets the VRML file and determines the meaning of its syntax, parsing generates built-in nodes and user-defined ones.

(b) These nodes are combined according to the scene hierarchy in order to construct a scene graph. Other mechanisms such as event processing by Routes or script interpretation are processed. The Browser also interprets the user's input and changes the viewpoint of 3D worlds based on the sensor nodes in ***VRML***.

(c) It generates audio/visual feedback to the user.



The next section discusses a proposed model of the multi-agent based system to enhance the navigation process in the Virtual Environment Browser.

4. Proposed model of the multi-agent in the Virtual Environment

The agent is the primary entity with the Virtual Environment Model, composed of capabilities, plans, databases, and events. Agents can address other agents and post events to them thus modeling inter-agent communication. We have setup the navigation assistant model as a collection of various agents, each with its own specialized capabilities. Each agent has a goal structure that governs:

(1) How they interact in the environment and
(2) The behavior of each agent.

A goal represents a task or states, which the agent is trying to achieve. Agents may have multiple goals and work to accomplish each one individually or concurrently with other goals. Each agent contains a goal list; these goals are sorted by order of precedence. The agents attempt to accomplish the higher precedence goal, and then move on the next goal. Determining which goals are active is based on the capability function of each agent and each goal is given a priority number.

4.1 The rules for building the multi-agent in the *VE*

The following steps are required to apply the agent model in the VE:

- **The Role:** The first thing we want to do is to figure out which role the agent should have in the synthetic Virtual Environment. This could typically be a description of the agent type, its personality and how it is supposed to react to the other objects.
- **Goals:** Some primary goals of the agent should be identified with the role description.
- **Behaviors :**These implements both goals and strategies to fulfill the goals.
  Behaviors are units which continuously compete to take control of the agent. They are organized in a hierarchy of groups containing mutually exclusive behaviors, where the top level group could be said to represent the agent's goals whereas the lower level groups represent different strategies to achieve the goal of their parent.
- **Drives:** Mostly an agent should have some sort of drives that act as an internal motivation to the agent and a bias to engage in a special line of action.
- **Information extraction:** The next thing to do is to identify the objects, agents and avatars in the world which the agent should pay attention to. Without these classifications, agent cannot assess any meaning to the sensory information.

The proposed model of the embodied agents in the synthetic ***VE*** consists of five agents. The first agent is a *Watching-agent*. This agent works as Interface Agent. The interface agent is usually not the interface between the user and the *VE*. Instead, it observes the interaction between the user and the *VE* from the sideline, learns from it and interacts with the controller agents through Headquarter Agent so we called it a Watching-agent. The second agent is a *headquarter-agent*, it works as a supervisor agent, and it is a mediator agent between the higher level agent (*Watching-agent)* and the lower level of the working agents. It stores the dynamically information about the *VE*. It sends this information to the working agents group. The third agent is the *Position-tracker agent*. It is used to determine the position of the navigator agents in the synthetic VE. It stores the status of each node, edge and path in the *VE*. The fourth agent is the *Object-identifier-agent*. It is used to keep track the changes of the dynamic objects in the *VE*. Finally, the fifth agent is the Navigator-Agent. It is the moving agent that helps the visitors of the *VE* to find their interested way. The navigator agent in the ***VE*** must be autonomous, deal with uncertainty, plan and decide what to do, react to unexpected situations that are; it has to overcome really hard problems if we want it to act in an intelligent and autonomous, thus navigator agent pose one of the biggest challenges for AI.

It solve Figure 4 illustrates the structure of the proposed Model of the Intelligent Agents embodied in *VE*.

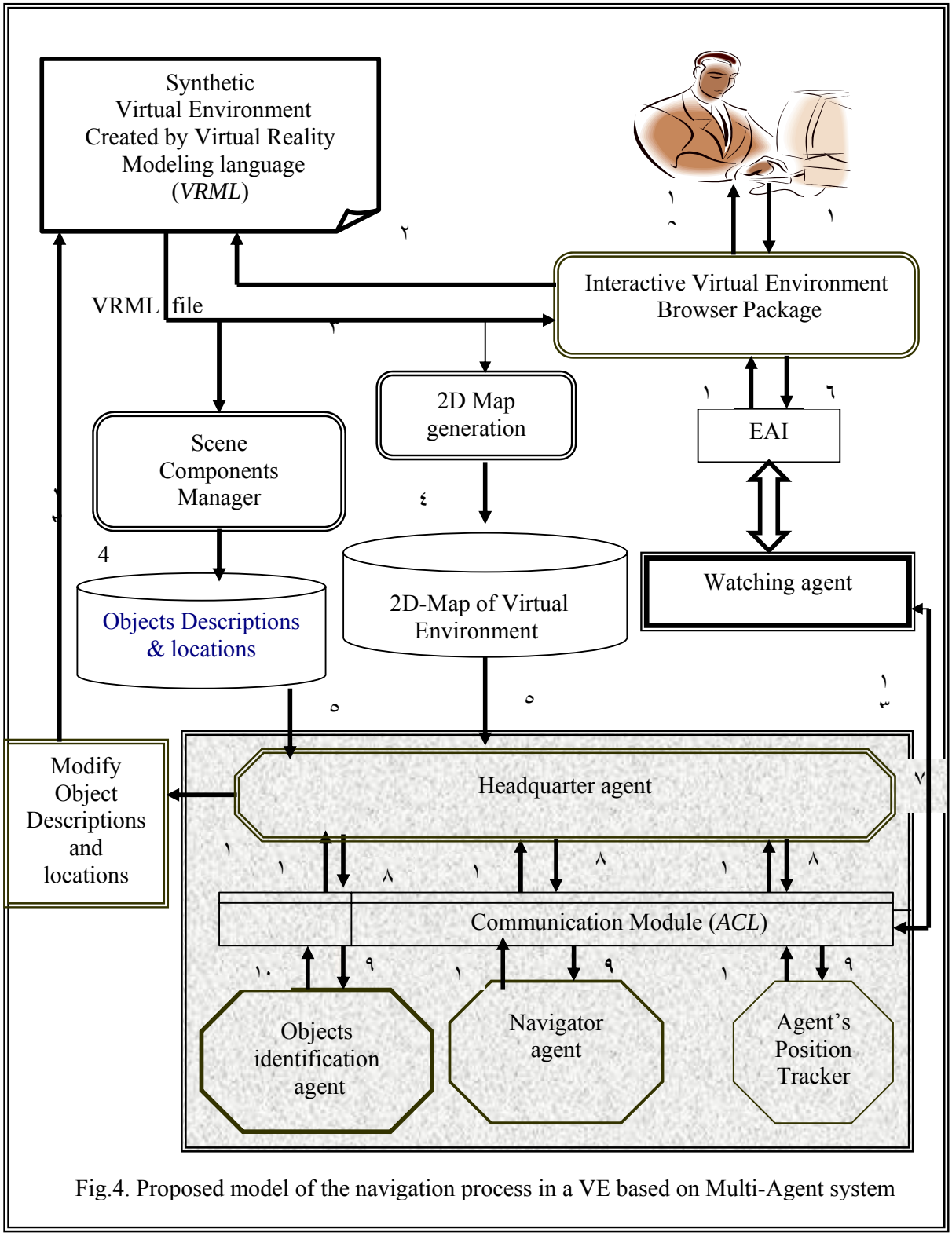


Fig.4. Proposed model of the navigation process in a VE based on Multi-Agent system

When the user loads the synthetic *VE,* the Browser reads it as a *VRML* file to display its contents at the same time the *VRML* file as input is used by 2D-Map extraction module. The 3D model of the environment is transformed into a 2D environment which is used as an aid by the *Navigator agent.* Also, ***VRML*** file is used by scene components manager module to build the database file which contains the static objects of the synthetic Virtual environment.

4.2 The 2D Map generation Module

The basic idea of this module is to determine whether a cell of the map should indicate the presences of a geometry that prevents navigation, by checking if the corresponding area in the *VE* can be traveled by the navigator agent created for the purpose. This is done by automatically moving a *ViewPoint* through the *VE* and detecting each collision of the viewpoint itself with any geometry. Whenever a collision is detected, the cell of the map that corresponds to the current position of the viewpoint is marked as containing geometry. We now describe in detail how the proposed approach is implemented.

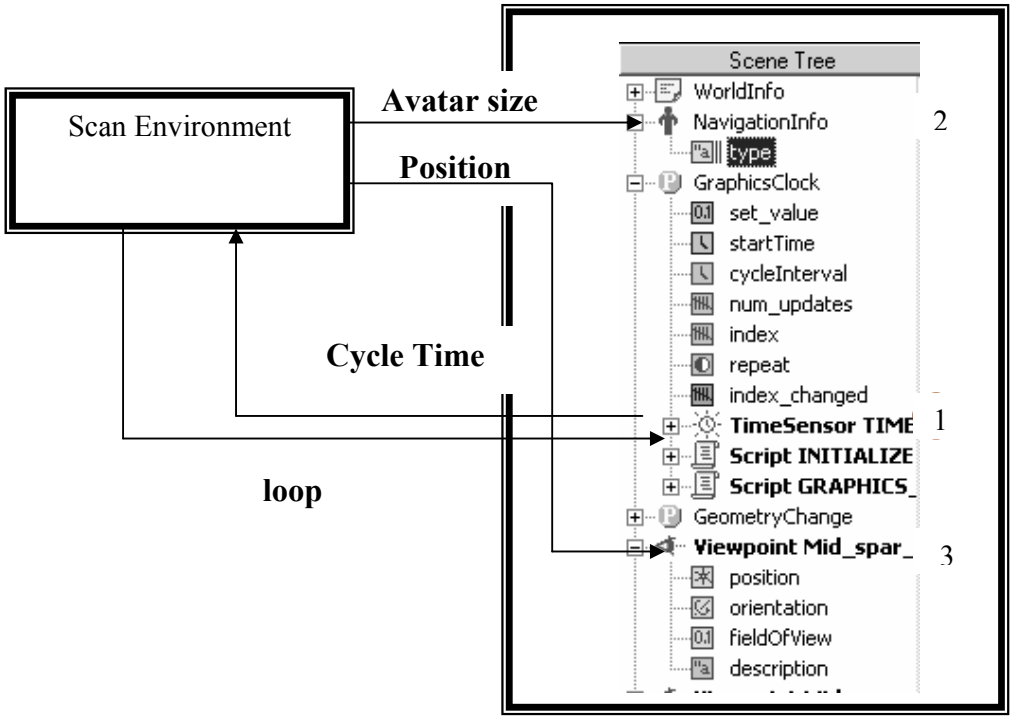


Fig.5 Description of the VRML nodes and routing of events for map extraction module

A *Time Sensor node* starts and stops the map derivation process, and controls the speed of scan. The derivation of the map is carried out by two VRML scripts, called *scanEnvironment* and *buildMap.* The first script computes and updates the position of the viewpoint to scan the entire *VE*. The second script receives detected collision events and updates the cell of the map that corresponds to the current position of the viewpoint. Collision events are generated by a Collision node, whose children include a *NavigationInfo node* (defining the size of the avatar used for scanning), a *ViewPoint node* (specifying the current position of the avatar).

4.3 Modify Object Descriptions and locations Module

During run-time, the environment is observed and all changes in the dynamic obstacles are tracked. Changes occur when obstacles appear, disappear, or change position. In all cases, two kinds of update operations are performed: cell occupation in case an object is detected to appear, and cell liberation in case an object is detected to disappear as shown in figure (6). Obstacle motions are treated with consecutive liberation and occupation operations. Cells are updated per object. Two sets are initialized , the set of objects disappearing and the set of objects appearing at a particular point in time. For each object, the cells occupied by the object are determined and sent to the modify object description and locations module. Therefore, each update operation receives one cell as input.

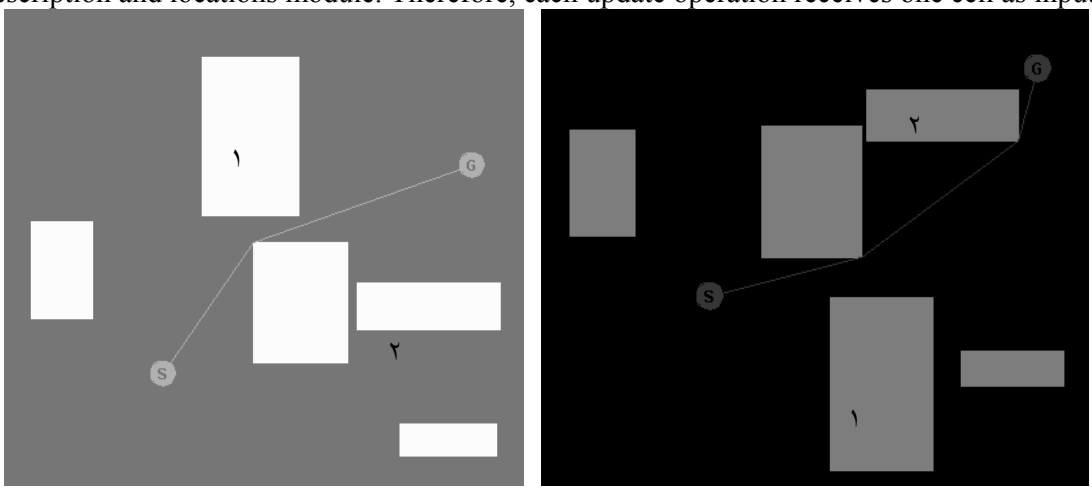


Fig.6. shows the path changes according to the different obstacles positions

4.4. Roles of the Embodied Agents in VE

Each agent has the capability function; it is the brain of each agent. Within the capability function, the agent takes into account its current knowledge about the environment, its active goals and the resources that it has to accomplish its goals.

4.4.1 The behaviors of the Watching-agent in the Virtual Environment

The *watching agent (*User Agent*)* is located at the position where the *VE* Browser is running, and which simply listens to the interactions of the user with the *VE* and sends relevant events to the *Headquarter-agent*. Accepting messages from the Headquarter-Agent, interpreting these and controlling the *VE* via the *External Authoring Interface, (*EAI). as shown in figure (7). *EAI* plays an important role in transferring the knowledge and the events between the Watching-agent and the user actions in *VE*. It is programming interface for communication between *Virtual Environment Browser* and *watching agent*.[6]

The Watching-agent layer consists of the *reasonable cor*e, the *Knowledge Base* and the abstraction knowledge of the *VE*. On the other hand, the *Browser* layer is also a composite module consisting of the *Static entities of VE* and the *Dynamic Virtual Objects Librar*y. It also contains containing also the agent's *Virtual Representativ*e (Avatar) and the VRML Central Processing Unit*, (VRML CPU).* The *reasonable Core* provides the agent with reasoning capabilities. Reasoning is supported by a number of knowledge bases which store various types of knowledge such as static and dynamic knowledge, domain knowledge, knowledge about the agent's capabilities, spatial knowledge about the virtual space, etc. The *VE* is

represented as a 'mental' structure in the agent's knowledge base, which is, in fact, an abstraction that maintains only the important information about it. The Logical Core reasons about the current world situation and according to its goals it sets some abstract actions which must be immediately executed at the virtual layer.

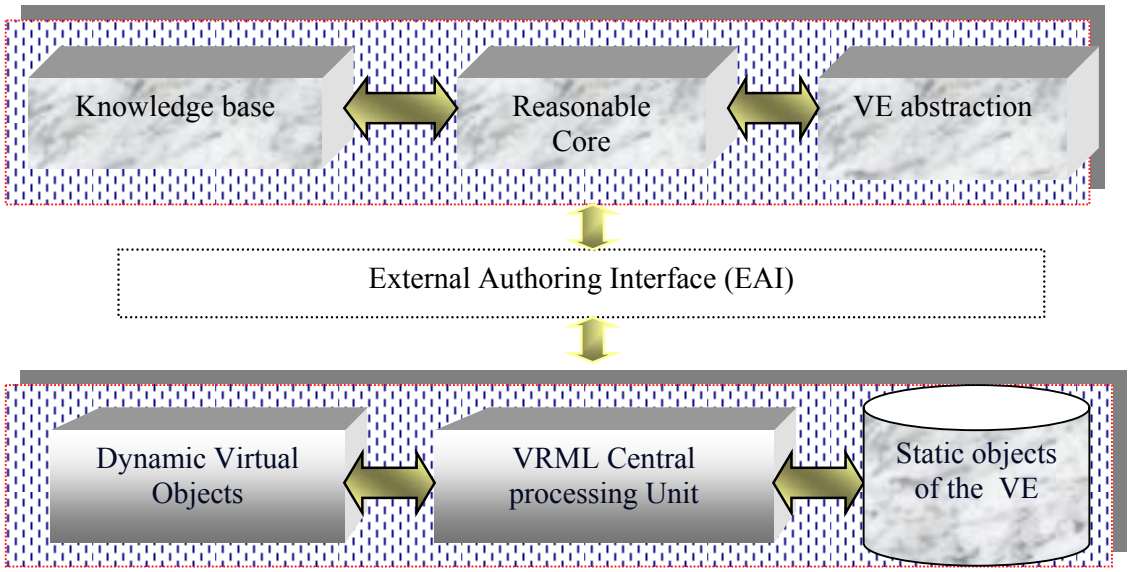
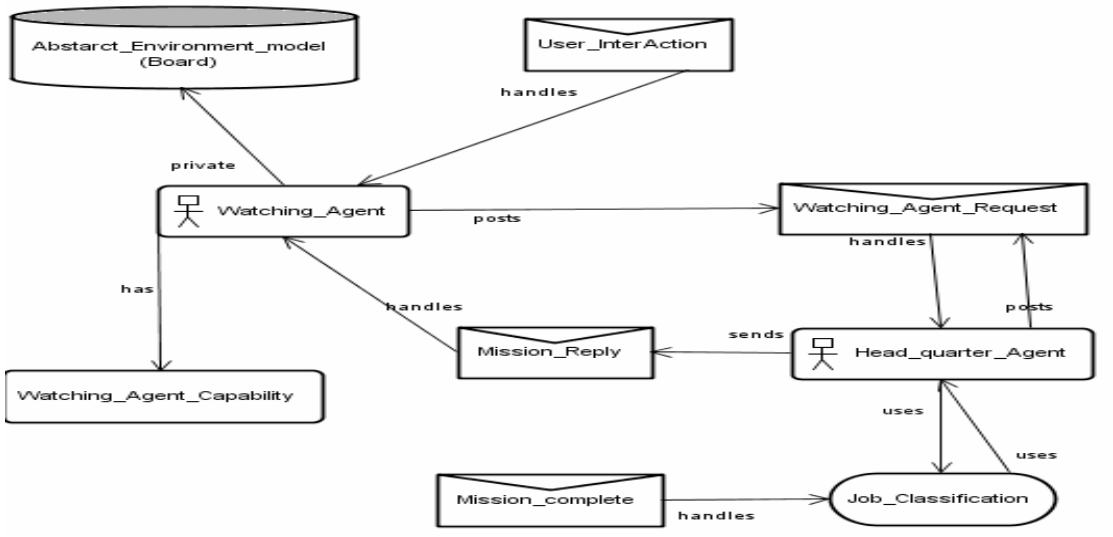


Fig.7. Watching-agent architecture as interface Agent of the VE Browser

The required actions, e.g. move to next location, are received by it. the action command arrives at the Virtual Reality Management Unit that specifies in detail the received actions. It provides specific values concerning the orientation and position of the avatar, e.g. it specifies the coordinates, orientation and path so that it can successfully move to the next location, and sends them as commands to the ***VE*** Browser. The Browser executes the command by altering the ***VE*** appropriately. When changes have been performed the ***EAI*** notifies the watching agent that the action has been successfully executed and the logical core goes on by updating its internal and external state. Consequently, the agent looks around into the virtual space, gathers any additional information and decides the next step it should take to achieve its goals.



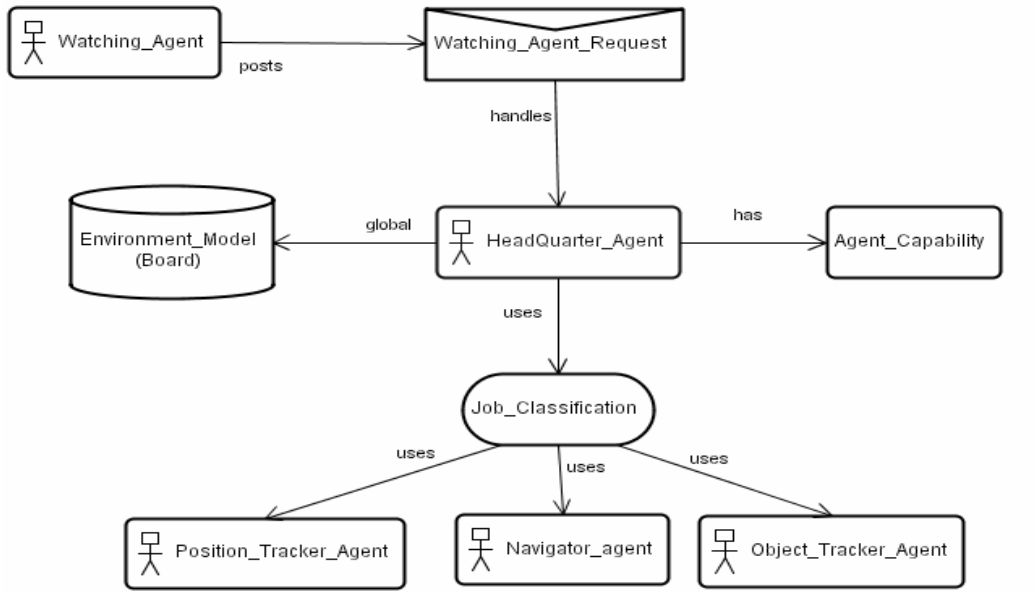
```
Has capability Watching_Agent_Capability cap؛
#posts event Watching_Agent_Request ev؛
#handles event Mission_Reply؛
#handles event User_InterAction؛
#private data Board Abstarct_Environment_model؛()
```

Fig.8. the model of the Watching agent's Behaviors

The Watching-agent responds to any changes occur in the Virtual Environments, traces any user's action and sends a request to the headquarter agent to tell what is the required actions. The headquarter agent handles the request by using Job-classification plan and decides which working agent is responsible for doing the required task. The Watching-agent is defined as a combination of a planner and a reactor.

4.4.2 The roles of the Headquarter Agent

1. *It creates*, *maintains*, *and stores* a user profile with all events history. Moreover, it controls a number of sub-agents that each focuses on a particular aspect of the user's behavior. Events coming from the *watching agent* are received by it, entered into the event history and then *send* to appropriate sub-agents.
2. It stores the previous communications with the user through the Watching-agent.
3. It keeps tracks of all the actions that currently are being performed (or have been performed).

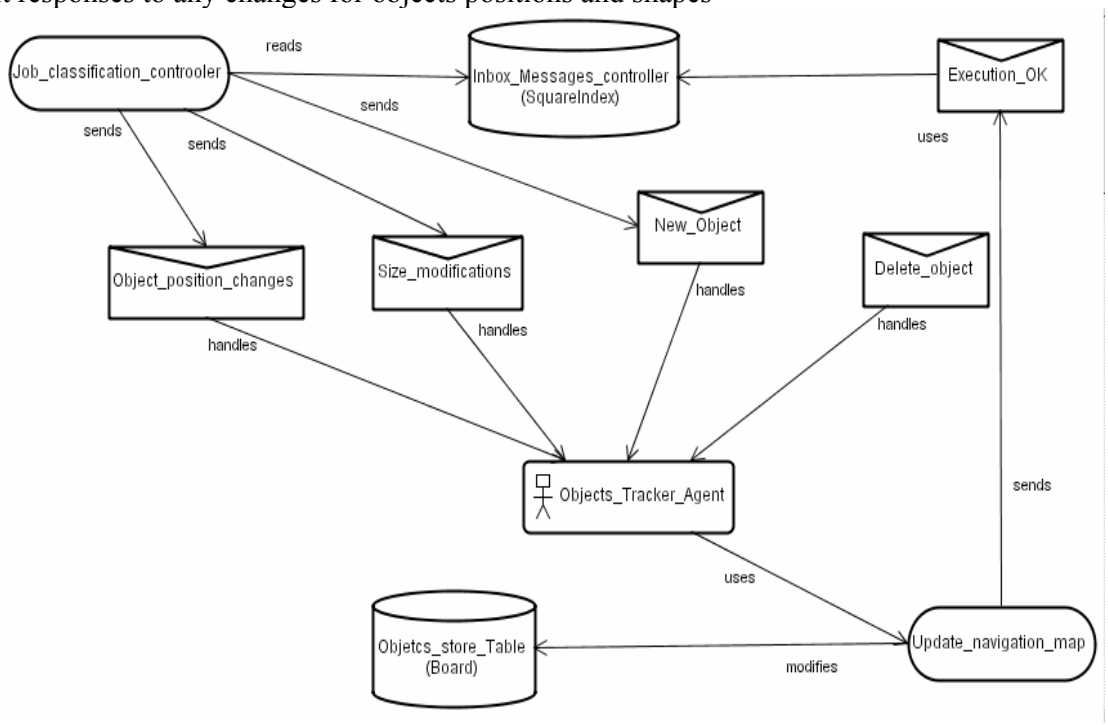


```
# handles event Watching_Agent_Request;
# posts event Watching_Agent_Request ev;
# sends event Mission_Reply ev1;
# uses plan Job_Classification;
# private data Environment_Data;()}
```

Fig.9. Headquarter agent's Behavior Model

4.4.3 The Roles of the Objects tracker Agent

It keeps track of the objects shapes and its locations.
It responses to any changes for objects positions and shapes




```
#handles event Object_position_changes;
# handles event Size_modifications;
# handles event New_Object;
# handles event Delete_object;
# posts event Reply_execution ev;
# handles event execution_reply;
# uses plan Update_navigation_map;
```

Fig.10. The Architecture of the objects tracker agent

4.4.4 The Roles of the Navigator agent

The Navigator-Agent is the main agent in the proposed model of the navigation agents in ***VE*** used to enhance the navigation process. The problem of intelligent navigator agent in ***VE*** is the subject of many recent AI researches. Even though many solutions have been proposed to address this problem, the ever-growing complexity of ***VE*** inhabited by sophisticated characters, makes it necessary to further elaborate computational models used for intelligent agent navigation. In this paper we develop our navigation algorithm based on the construction and use of the so-called *navigation map*. The algorithm is perfectly suitable for the use in modern VR systems, 3D video games and simulators. The Navigator Agent is capable of pursuing its goals (that might have been specified by the users) on their own. This relieves the user from technical detail, e.g., specifying obstacle avoidance strategies. The Navigator Agent helps the user to find specific locations in the *VE*. It is specialized in determining appropriate free and safe paths through the VE, and can provide guidance to the user that tries to follow such paths. It computes possible free paths. Information required by the navigator-agent are:

- The navigator agent should be able to reason about the geometry of the world in which it moves. It knows about the user's coordinates in the ***VE*** and it has knowledge of the coordinates of a number of objects and locations. This knowledge is necessary when a user refers to an object close to the navigation agent in order to have a starting point for a walk in the environment and when the visitor specifies an object or location as the goal of a route in the environment. The navigator agent is able to determine its position with respect to nearby objects and locations and can compute a walk from this position to another with coordinates close to the goal of the walk.
- The navigator agent knows more about current position and focus of gaze of the user, geometric relations between objects and locations, knowledge of previously visited locations or routes and knowledge of the previous communication with the visitor.
- *Interest Points*: positions where the agents can go. Each Interest Point is surrounded by a set of points that form a limit region around it. This region is used to distribute the agents in the case of multiple-agents navigation. These region points can be connected within themselves, forms access regions between one Interest Point and another. The connection forming graphs that represent the possible paths to be applied by the agents. The paths are formed by the connection of two Interest Points. Once there is a path between two Interest Points, the navigator agent is able to apply this path.
- Navigation map generation means the automatic generation of paths to link up two or more Interest Points.
- Criteria specification associated with the paths that could be used by the agents during the navigation process.

Fig.11. illustrates the relationship between the navigator agent and the navigation module. The navigator agent precepts and receives an initiation massage from the headquarter agent and sends a request to the navigation module. the navigation module is responsible for leading the navigator agent from a source location to a destination, avoiding danger and obstacles. After the navigation module completes its plan, it sends a safe path to the navigator agent to follow it.

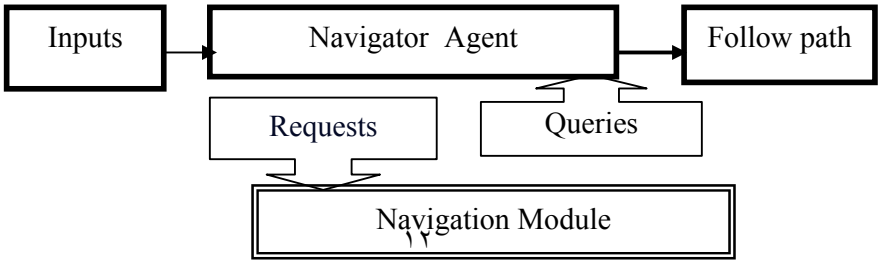
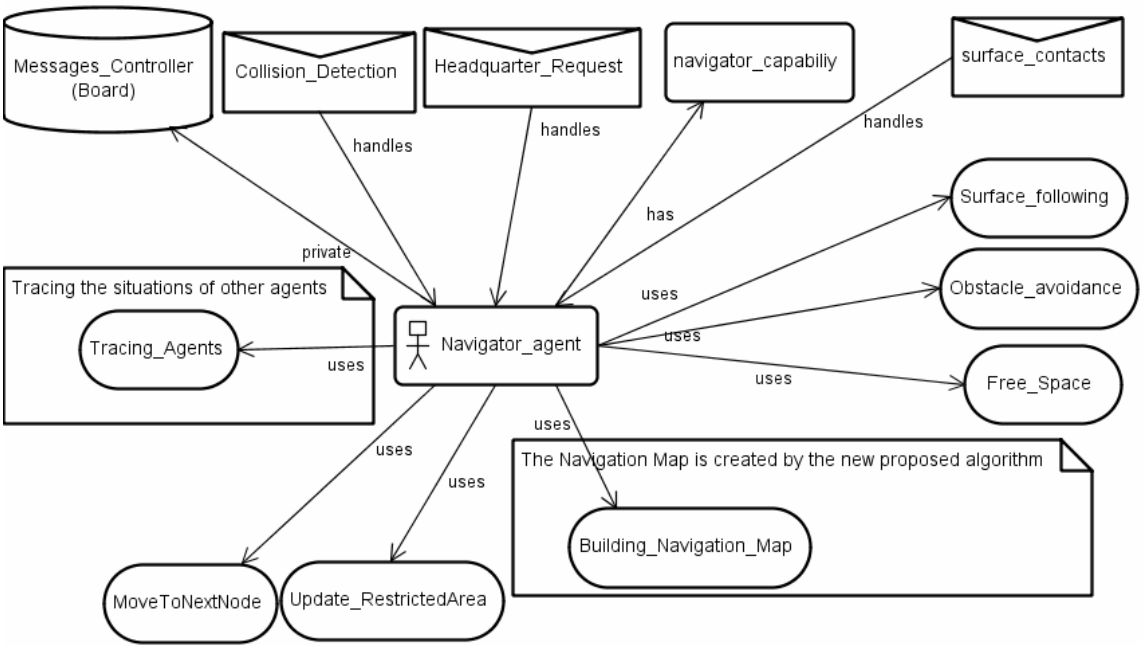


Fig.11. relationship between the navigator-agent and the navigation module




```
import moving_agents.Tracing_Agents
public agent Navigator_agent extends Agent {#has capability  navigator_capabiliy cap
    #handles event Headquarter_Request
    #handles event Modification_2Dworld
    #handles event surface_contacts
    #uses plan RestrictedArea
    #uses plan Free_Space
    #uses plan MoveToNextNode
    #uses plan Building_Navigation_Map
    #uses plan Tracing_Agents
    #uses plan Surface_following
    #private data Board Messages_Controller()
```

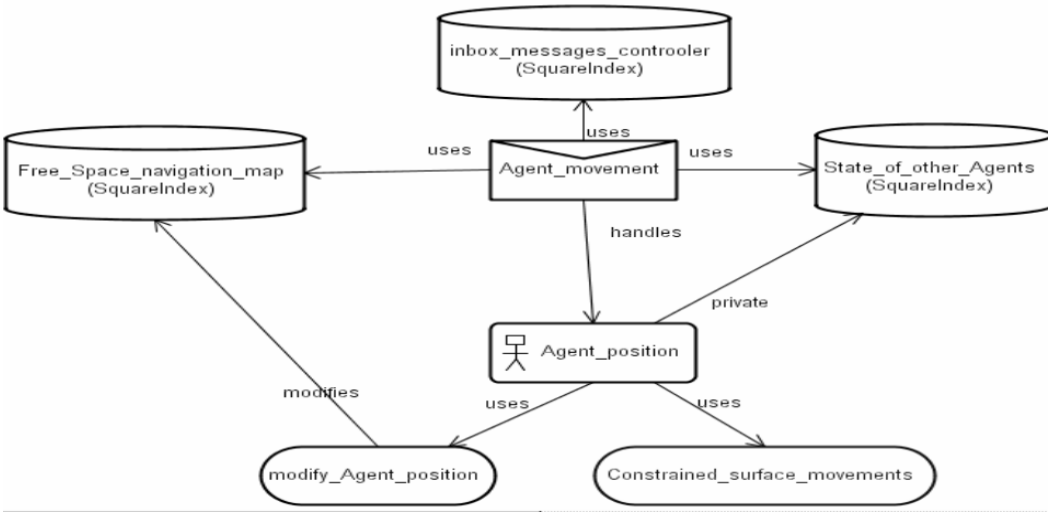
Fig.12. The Architecture of the navigator agent

Navigation module plays an important role in formulating proper paths to reach a destination point. Often the agent must deviate from its intended course to negotiate obstacles that are in its path. After the maneuver has been made, the agent may find that it has strayed in a direction farther away from it goal. In this case the agent must take corrective actions to move towards its goal. A fitness function may be used to determine how well the agent is meeting its goal of navigating to an assigned waypoint. The Navigator-Agent communicates with the

headquarter-agent by using KQML languages. The following messages to the headquarter-agent that define the agent's actions in the environment:

4.4.5 The roles of the Position Tracker agent

1. Maintaining knowledge of current agent's position and orientation is frequently problem for a people in the VE [2]. The solution of this problem is a position-tracker agent. The *Position Tracker* Agent maintains knowledge about the position of the navigator agents which is dynamic as it changes over time
2. Monitors the current position of the navigator agent, looks for the next view and keeps track of the previous locations and what is within the eyesight of the visitor. It is a very useful method when the environment contains multiple navigator agents
3. □t keeps track of the previously visited locations by the user.




```
public agent Agent_position extends Agent {      #handles event Agent_movement
    #uses plan modify_Agent_position
    #uses plan States_of_other_Agents
    #uses plan Constrained_surface_movements
    #private data SquareIndex State_of_other_Agents()
    public Agent_position(String name)
```

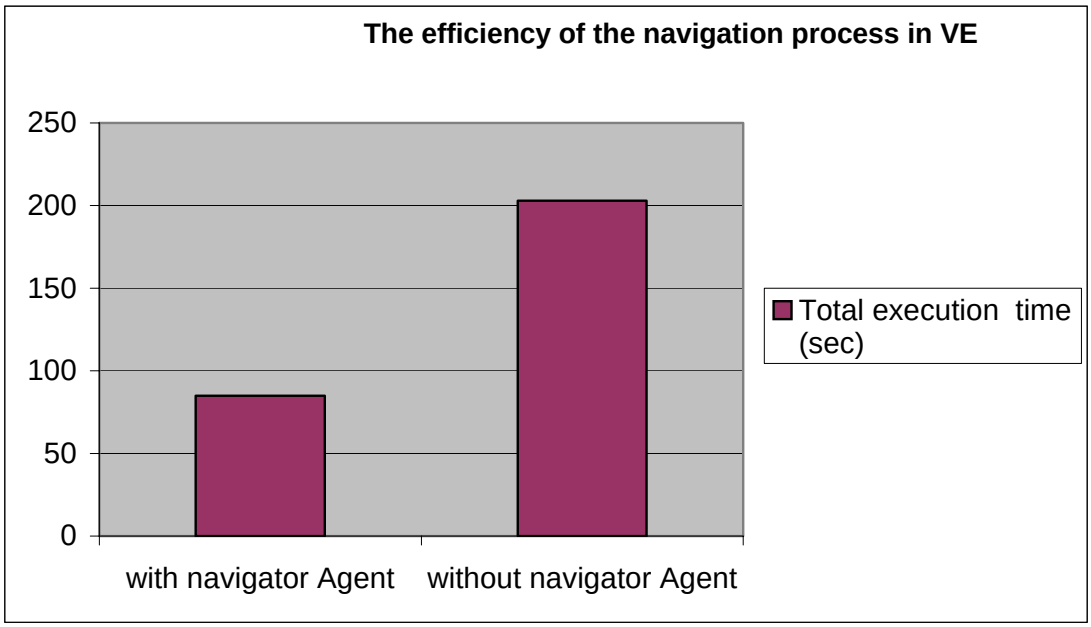
Fig.13. The architecture of the position tracker agent

5. Experimental Results

The experiment is done to measure the efficiency of the ***VE*** Browser after we applied the proposed model. In order to compare the effects of incorporating navigator agent into the VE Browser, the experiment consists of two runs (with and without navigator agent). The results of this experiment are summarized in Table (1). The overall times taken to complete the requested task are 85 and 210 seconds, respectively for the VE with and without navigator agent. The performance speedup for the ***VE*** with navigator agent is about 42 %. In term of movement steps, about half of the steps are saved after the navigator agent is used.

Table (1) the efficiency of the navigation process in the VE Browser with and without navigator agent

Comparison points	with navigator Agent	without navigator Agent
no of navigation steps	584	1025
Total execution time (sec)	85	203



6. Conclusion and future work

The combination of intelligent techniques and tools, embodied in autonomous agents, together with effective means for their graphical representation and interaction of various kinds, have given rising to a new area at their meeting point, which we call Intelligent Virtual Environment. The architecture of an intelligent navigation agent system is presented and applied in the synthetic Virtual Environment. The synthetic Virtual environment embedded with autonomous agents offer a general solution of shared objects and autonomous agents in Virtual Environment. Interesting behaviors such as obstacle avoidance or exploration following a given stimulus towards a goal are easily expressible in this environment and opens up for a variety of biologically-inspired behaviors for virtual agents populating the virtual environment. The performance of the navigation process in the ***VE*** is increased by ***42***% when the embodied agents are added. For the future work, Developing an inter-agent protocol that will enable the agents to communicate with each other and exchange information, in order to accomplish given tasks faster than using the current communication languages. Extend this model to be able to solve the problem of the multi-navigator agents in shared ***VE***. more experiments need to be carried out for different tasks, on different ***VE***, and on different systems of various computing powers.

References

[1] Chan Su Lee and Grigore C. Burdea., Virtual Reality Technology, Second Edition Laboratory Manual, The State University of New Jersey U.S.A, John Wiley &Sons, (2003).

[2] Dzmitry Aliakseyeu, Sriram Subramanian, Jean-Bernard Martens, Matthias Rauterberg: "Interaction Techniques for Navigation through and Manipulation of 2D and 3D Data", Eigth Eurographics Workshop on Virtual Environments,W. Stürzlinger & S. Müller, (2002).

[3] Tim Batchelor Hnd., ANTS: Automatic Navigation of Terrain Systems, Alumnus of Bolton Institute, (2003).
[4] L. Kavraki, P.Svestka, J. Latombe, and M. Overmars., Probabilistic Roadmaps for Fast Path Planning in High-Dimensional Configuration Spaces., IEEE Transaction on Robotics and Automation., 12:566-580, (1996).
[5] Luca Chittaro, Lucio Ieronutti, Roberto Ranon., Navigating 3D Virtual Environments by Following Embodied Agents., PsychNology Journal., Volume 2., Number 1, 24 – 42.,(2004).
[6] Rikk Carey, Gavin Bell, Chris Marrin., Virtual Reality Modeling Language., International Standard ISO/IEC 14772-1., The VRML Consortium Incorporated., (1997).
[7] Ruth Aylett and Marc Cavazza., Intelligent Virtual Environments-A State-of-the-art Report, University of Salford, CVE, Salford., University of Teesside, School of Computing and Mathematics, (2004).
[8] Slater, M., Usoh, M., and Steed., A Steps and ladders in virtual reality., In ACM Proceedings of VRST., 45–54. (1994).
[9] Tsai-Yen Li, Hsu-Chi Chou., Motion Planning for a Crowd of Robots., in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)., (2003).
[10] Wenfeng Li, Henrik I Christensen, Anders Oreback., An Architecture for Indoor Navigation., Proceeding of the International Conference on Robotics & Automation IEEE, p1783-1788., (2004).