EG UK Theory and Practice of Computer Graphics (2005)
L. Lever, M. McDerby (Editors)

Simulation and Visualization of a large scale Real Time Multi-Robot system

G. Al-Hudhud & A. Ayesh & H. Istance[†]

De Montfort University
and
M. Turner[2‡]
The University of Manchester

Abstract

This paper describes the software implementation and the visualization aspects of an interaction-communication protocol within a large scalable multi-robot system. It investigates the current communication protocols within multi-agent systems and the feasibility to transfer them into a virtual environment system that performs a specified task intelligently, embedding human capability into the control system software. The proposed system allows dynamic changes, i.e. the user may be able to continuously issue commands, or modify tasks. The work presented exploits the Virtual Environment Centre VEC. The semi-immersive full scale environment within the VEC allows the user to better understand the robots' behaviour and in turn test whether they simulate the expected behaviour. The use of a semi-immersive full-scale environment also gives an increased level of presence enabling the user to believe they are within their own simulation. It also presents a prototype for a robot automatic fire extinguishing system as a test application area.

1. Introduction

Virtual environments are considered a powerful tool for use in realistic simulation studies within different applications. Among these, robotic applications are of significant interest. For example a simulation of a multi-robot system can be evaluated and assessed by operating in a simulated environment where the simulated robots can be exposed to a variety of different tasks and surroundings without an excessive amount of development time. Thus, cooperative multi-agents system can be developed and used as a testbed for higher-level tasks without the necessity of developing the specified hardware or transport the agents physically to the location. For example, the same agent structure can be applied to tasks involving outer space or undersea exploration.

In the real world, cooperation in turn requires a sufficient amount of communication that appears as a series of social activities: negotiation, dialogues, interaction. Cooperative agents communicate to exchange information in order to perform a specified task. Such large groups usually are faced with a high degree of complexity, i.e.dynamic and unpredictable environments. Hence, controlling the movements of such organization is of great importance as most of the performed actions by these agents must be guaranteed within real-time and scalability constrains.

In order to develop a communication protocol within this large multi-agent system it is essential to specify in detail the motion control techniques used as well as the mechanism used to interactively specify tasks to agents. Finally, it is important to understand and assess the simulated behaviours at all developing processes. Therefore, a major interest of this work is to present both the specified communication techniques and the visualization tools used to evaluate the system performance.

Section 2 discusses related work of controlling the movements of a large scale multi-agent system. In section 3, the mathematical method concerned with both local and global interaction with the world is discussed. Section 4 discusses the real time system used to visualize and assess the system

[†] Centre for Computational Intelligence& Virtual Environment Centre

[‡] Manchester Visualization Centre, Manchester Computing

performance and the advantages of the proposed visualization tools over the pure simulations. Section 5 presents the preliminary results of the experimentations.

2. Related Work

2.1. Cooperation within a Multi-Agent System

In order for a set of agents to cooperate and perform a specified task intelligently, they must implement some form of social interactions. Communication and organization are considered to be the mainstays of cooperation [BdL*98]. Therefore, organization and communication concepts are considered to control the autonomously moving geometrical objects within the simulated environment, in order to exchange information about goals, intentions, results, and the state of other agents.

An example of cooperation between agents is a cleaning task that requires a pre-defined path planning to cover the unoccupied areas in the specified environment [LYS03], which is an essential issue in robotic applications, e.g. searching tasks. For relatively small scaled systems, this has been shown to be feasible.

So, in large scaled systems that represent complex environments with hundreds of moving objects, the conventional pre-specified paths become impractical. Also, in a situation where the environment is not static, where all objects might be in motion themselves, it is simply not practical to try to fully specify all paths beforehand. A key issue when trying to develop a large organization of multi-agents is to overcome the problem of organizing themselves efficiently. Hence, complexity of communication and the risk of failure within such systems has raised the need for the evolution of behavior that synchronizes the dynamics of such large groups.

2.2. Motion Control Techniques within a Large scale Multi-Agent System

Regarding the motion within a dynamic and unpredictable environment, adaptive movement behaviour techniques have developed a long way within computer graphics. Early models involved the movement of a large crowd on large scale terrain [TWP03], which require an accurate environment map before the path planning algorithm is executed, as it is essential to pre-specify paths around a static environment. Systems capable of complex planning of partially unknown environments [WCE03] have used centralized solutions which is considered to be impractical when developing a cooperative multi-agent system [GM01].

2.3. Visualization

Researchers in the field have addressed a number of problems using the conventional simulations, for example the time required to run the simulations as well as objects were represented by static geometrical objects which move according to pre-exisiting plans. Within this paper, the second concern was to develop a real time system within a virtual environment. The simulation process involved modeling the environment and the agents. Originally agents within the simulation were represented as abstract shapes, just indicating the minimal information required of location and heading. The use of complex robot models for increased resolution as can be seen in the figures in section 6, were introduced, even if not accurately representing real robot design. The key issues will be discussed in section 5.

3. Methodology & Technique

The proposed technique requires agents to posses a local and global knowledge in order to communicate and coordinate their movements and actions [NLJ94]. Regarding movement coordination, a directed flocking behavior [Rey87] is used to organize the group as a system for producing animated simulations for animal behavior via three interaction rules. A recent example of implementing this motion control technique is described by Watson *etal* [WJC03]. In this situation, agents only have local views, goals and knowledge.

The proposed model extends the flocking algorithm used in [WJC03] to establish a higher level of communication with a human actor to perform specific tasks. Therefore and to avoid the disorder that results when agents no longer possess a global view of the entire organization, agents have to globally communicate with a human actor, agents then cooperate through a global blackboard [NLJ94] as a fourth rule. The three flocking rules and the blackboard communication rule form the set of intelligence rules an agent uses to move and interact with its environment.

The agent's internal system is composed of four main components: sensor, communication, interpreter, and a metalevel component. Each agent continuously updates its knowledge every time interval $\tau = 1/Framerate$ by constantly reading both the sensory data and the blackboard message and replacing old information with new information. The agent places the captured information in its Knowledge Representational System (KRS). Once the information is passed to an agent's internal system, the interpreter tests this information against a set of intelligence rules and performs the numerical computations over the time interval (τ). An agent revises its beliefs according to the new information placed in its Knowledge Representational System before deciding on the next action which is most probably 'correct and in a real time fashion'.

3.1. Intelligence Rules

The proposed communication algorithm exploits the flocking rules, previously described by the authors in [AHTA04] and [AHATI04], and the blackboard messages rule. The

flocking rules are used to control the movements of a large group of agents and consider the sensory data as the only source of an agent's knowledge. Consequently, the agent's movement is partially characterized by a sensor. The blackboard messages represent the final part of the agent's knowledge and forms the fourth rule in our communication algorithm. The flocking rules in addition to the blackboard rule form the set of intelligence rules that provides an agent with sufficient intelligence to move autonomously within its environment. These rules are described as follows:

1. **Alignment Rule** R_α
   According to the sensory data, each agent A_i searches the nearest agent A_j and tries to align its velocity vector with the velocity vector of A_j. The velocity vector (centroid $C_{A_i}^{R_\alpha}$, correction angle $\theta_{A_i}^{R_\alpha}$) obtained from this rule is used to modify the current heading ψ_{A_i} of agent A_i. A weighting value associated with this rule is $w_{A_i}^{R_\alpha}$. $w_{A_i}^{R_\alpha}$ is used to adjust the strength of the alignment force.
2. **Cohesion Rule** R_β
   This rule acts as a binding force where each agent tries to orient its velocity vector toward the centroid of the detected team members. The velocity vector (centroid $C_{A_i}^{R_\beta}$, correction angle $\theta_{A_i}^{R_\beta}$) obtained from this rule is also used to modify the current direction ψ_{A_i} of agent A_i. A weighting value associated with this rule is $w_{A_i}^{R_\beta}$. $w_{A_i}^{R_\beta}$ is used to adjust the strength of the cohesive force. An agent's field of view for both alignment and cohesion rule is *360 degrees*.
3. **Collision Avoidance Rule** R_γ:
   This rule is essential to avoid overcrowding as well as colliding with other objects(static or dynamic). According to this rule and when agent A_i detects an agent A_j, it can do one of two actions: move away from agent A_j if it is too close, or move toward it if agent A_j is far away. An agent's field of view for this rule is *36 degree* ahead. The velocity vector (centroid $C_{A_i}^{R_\gamma}$, correction angle $\theta_{A_i}^{R_\gamma}$) obtained from this rule is used to modify the current direction ψ_{A_i} of the agent A_i. The weighting value $w_{A_i}^{R_\gamma}$ is used to give the priority to this rule over the other rules.
   The collision avoidance procedures used in our algorithm exploits the sensory data stored in three locations only. These locations are the straight ahead and 18 degree both to the left and to the right. This implies that the field of view is specified by 36 degrees ahead and within a predefined minimum separation distance between a robot and other objects. If a 3D-object exists in this field of view, it is detected at least by one or more of the three lines of sight that cover this field. The detected object is avoided by testing its position against a set of a predefined cases. These cases are: (a) if the detected object is straight ahead then change heading 18 degrees to the left and move provided that no objects exist 18 degrees to the left otherwise the change in the heading is 18 degrees to the right. (b) if the detected object is located 18 degrees to the right and change the heading 18 degrees to the left then move provided that no object exists straight ahead. (c) if the detected object is located 18 degrees to the left, change the heading 18 degrees to the right then move provided that no object exists straight ahead.
4. **Blackboard messages rule** R_{BB}:
   It utilizes message passing via the blackboard that allows agents to communicate with a human actor to perform a specified task. An agent A_j computes a velocity vector (centroid $C_{A_i}^{R_{BB}}$, correction angle $\theta_{A_i}^{R_{BB}}$). The velocity vector produced by this rule depends on the position of the target. The weighting value $w_{A_i}^{R_{BB}}$ is used to adjust the strength of this rule.

How does an agent makes its decision? After the interpreter calculates the velocity demands from the four interaction-communication rules, it vector-sums the velocities each frame according to the following equations:

$$X_{A_i}^{\tau} = \sum_k w_{A_i}^{R_k} C_{A_i}^{R_k} \cos\left(\theta_{A_i}^{R_k}\right) \tag{1}$$

$$\Upsilon_{A_i}^{\tau} = \sum_k w_{A_i}^{R_k} C_{A_i}^{R_k} \sin\left(\theta_{A_i}^{R_k}\right) \tag{2}$$

From equation 1 & 2 the new heading angle ζ_{A_i} is calculated:

$$\zeta_{A_i} = \arctan\left(\Upsilon_{A_i}^{\tau}, X_{A_i}^{\tau}\right) \tag{3}$$

Finally, the agent's metalevel component tests the appropriateness of the next action and the suitability of the decision that has been made. Accordingly, agents process the information in two steps: first, where the interpretor decides on the next action. Second, where the metalevel component tests the next action against a set of predefined undesired situations, and prevents the system from reaching any of these undesired situations.

4. Software Development

The real-time software used can be described in two parts: (a) Representing the scene and the objects attached to it. This implies creating 3D models and has been done using a 3D modeling package (MultiGen Paradigm Creator). (b) Simulating the interaction between agents and rendering. The software environment VegaPrime is used for the real-time visual simulations including the scene description, interacting with scene, visualising interaction between agents within the environment, and allowing testing the algorithm in real-time.

The user interface for Vega Prime is Lynx Prime where most of the set up is managed. The output of Lynx Prime is a VegaPrime Application Configuration File (ACF). ACF is the input for the application which is composed of Vega Prime API and integrated C++ within which all the intelligence rules that an agent need to move and interact with the environment are defined, see figure 1. The runtime

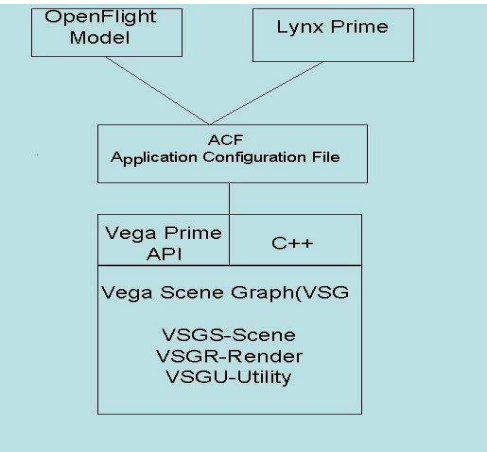

Figure 1: *Vega Prime System Architecture, [Ant03].*

control then includes:(a) defining the ACF. (b)configuring the ACF and the system.(c)executing the runtime loop which contains the user's methods. (d) shutting down at the end of the application. In a VegaPrime application, after initializing the system the next step is to create a VegaPrime instance using VegaPrime application class defined in (vpApp.h).

```
Vega Prime Application: Runtime Control.
#include< vpApp.h >
int main(int argc,char * argv[])
{
// initialize vega prime
vp::initialize(argc, argv);

//create a vpApp instance
use the class has been created
myApp *app = new myApp;

// load acf file, assumes argv[1]is the acf file
app→define(argv[1]);

// configure the application
app→configure();

// runtime loop: the run() method executes
the main simulation loop
app→run();

// unref the app instance
app→unref();

// shutdown vega prime
vp::shutdown();
return 0; }
```

4.1. Realism of the Simulation

The realism of the simulation is critical to ensuring that the implemented algorithm during the simulation is feasibly transferable to a real application e.g. robots. The real-time 3D simulation features of the Vega Prime again provide the chance to focus on the specifics of the application to satisfy special simulation needs, e.g. laser sensors. This aims at giving a deeper understanding of the simulation and a possible easier introduction to seeing how a real life robot system would be employed. The sensor for any agent is implemented as a collision detection of a fixed line segment at a defined angle centred just outside of the agent's location. The length of the line segment determines the laser sensor range. The sensor rotates 360 degrees quantized into 18 degree intervals, therefore it detects one of 20 locations each frame. The horizontal sensor fired from the robot toward the scene returns the calculated distance between the robot and the intersection point on the detected object. Also, at each location the sensor returns the detected object's identity location and heading. For each frame, the agent's sensory data is stored in (1×20) array. The information stored here are filtered as each rule focuses on the relevant information only. This allows an agent to easily recognize its team members, remembers the other detected objects, their directions and their identities. Figure 2 shows that at each location, the sensor detects only one object. For example in figure 3 there are two objects that fall on the same line of sight, the sensor will only detect the first hit one and will not detect the other (object A in figure 3).

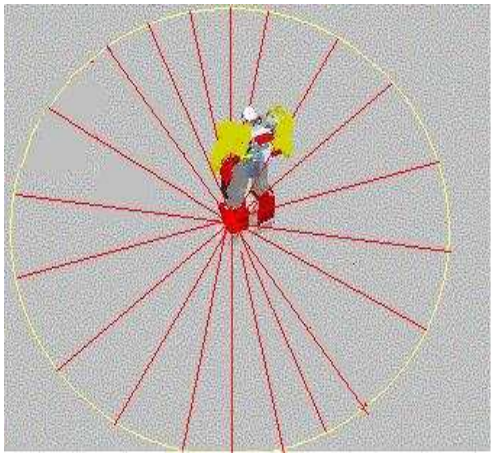

Figure 2: *Rotating Sensor: the red line represents the sensor.*

The collision avoidance procedures used here allow for smooth change in the robot's heading and also allows the robot to steer around the obstacle rather than stopping and then change heading. In the case where the agent's sensor detects three objects in the three locations then it neither changes the heading nor moves until the next frame comes

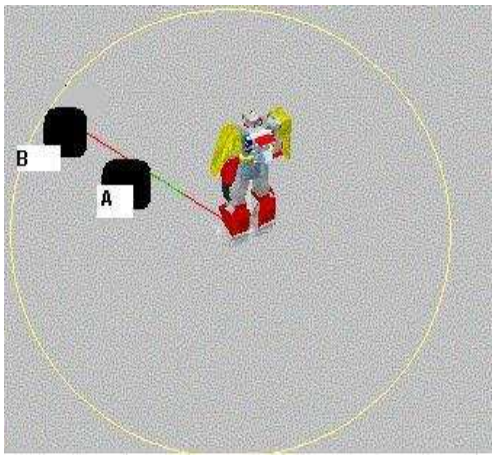

Figure 3: *The sensor renders green once it hits an object. Object A is detected while object B is not.*

with a new sensory data that may show a new set of detected objects and locations. If the new information keeps showing detected objects in the three locations this means the agent is "dead" and can not move anymore.

By using standard hierarchical bounding volumes dictated around all the robots and other objects the complexity of the collision detection algorithm can probabilistically be reduced from a worst case scenario of $O(n^2)$ as the number of agents increase. In practice, as shown in table 1, a linear scale-up has been observed for up to 50 agents.

4.2. Rendering

The number and position of the observer, how to move around the scene and the specification of the environment are managed by LynxPrime. Considering the observer as an eyepoint or camera, everything drawn in the drawing area including the scene and the attached objects is drawn from the observer's perspective. Scene display has then been managed by both defining the level of details and setting a number of eyepoints. The level of detail allows rendering a collection of robots upon their distance from the eyepoint. Typically, higher detail geometry is rendered when objects are close to the eye point and lower detail geometry is rendered when objects are further away. It also allows for smooth transitions between these levels of detail, where there would otherwise be abrupt changes. Using this approach, higher detail is only rendered when it is close enough to the eye point that the detail can be seen, and progressively lower detail is automatically used as the eye point gets further away. This increases the appearance of scene complexity whilst reducing the cost of always rendering the highest detail.

4.3. Visualisation of Agents' behaviours

A cluster of 6-PC's is used to generate 6 images that are projected onto a *145 degrees* curved screen (giving a $3K \times 1K$ resolution). The agents' movements and interaction inside the virtual world are projected on a big screen in the VEC. All the camera views can be drawn and are displayed at once on the screen allowing multiple users to visualize and assess the simulation. There is complete flexibility of viewing location and direction allowing an observer to monitor and view any action possibly with multiple views in different windows; including for example both global and agent view points. The big advantage over a pure simulation is the building of a semi-immersive full-scale virtual environment that mimics to different quality levels the real physical space. The use of a semi-immersive full-scale environment, which the VEC allows, gives an increased level of ŚpresenceŠ enabling the user to believe they are within their own simulation. This allows for real-world 'what-if' problems to be tested and seen in an easy to understand manner. Group and collaborative analysis can then also take place in the Centre allowing experts from various fields to monitor, discuss and control a simulation.

Figure 7 shows an abstract version of the the VEC at De Montfort University. The space in front of the screen hosts up to 20 users at once. The users are able to evaluate subjectively the simulated expected behavior. The experiments' expectations are designed to be able to show emergent behaviours in environments with hundreds of agents. The choices made for the visualization allow the user to better understand the way agents move inside the world and how they make their decisions.

5. Experimentation Test

For the purpose of testing the described algorithm, a series of experiments have been carried out. The initial position and direction of the agents were set randomly but were placed inside navigable areas. The initial experiments were designed to test both: the optimum sensor range that satisfies locality needs for our agents. The results showed that as the sensor range increases the agents are not capable in groups to pass through narrow areas between the two obstacles, figure 4.

Also, the task completion time increases as an agent takes time to steer around obstacles instead of passing through possible narrow areas between them. Finally, a fire extinguishing task has been assigned to a set of agents and the resulting behaviour has been evaluated. This experimentation has been developed to test the following two models:

5.1. Local-Global Communication (LGC) Model

Agents in this model communicate both locally and globally, figure 5. Agents communicate with each other and with

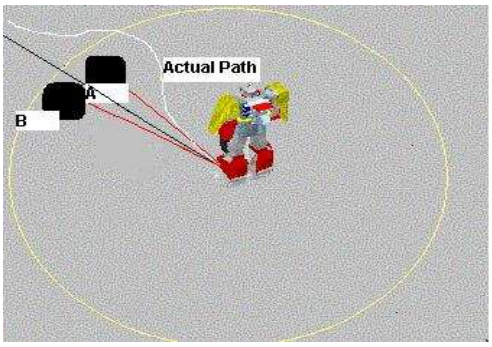

Figure 4: *The agent can not pass through the two Objects A & B. The white line shows the actual path*

a human actor. Agents are grouped in teams, as tasks are allocated as team tasks. Applying the proposed algorithm increases the chance of task completion despite any difficulties that may arise. For example, when any agent fails to perform the specified task, other team members will continue to perform the task. Figure 6 shows a set of agents circling around the fire location.

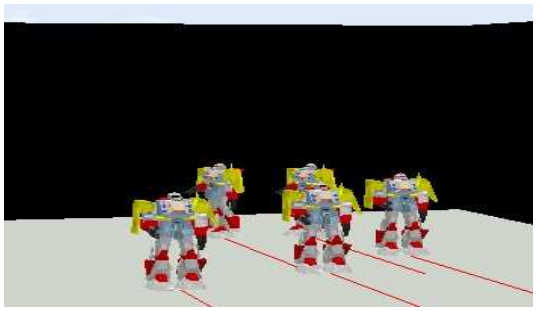

Figure 5: *A set of agents moving within a team and influenced by the cohesion force and the alignment force.*

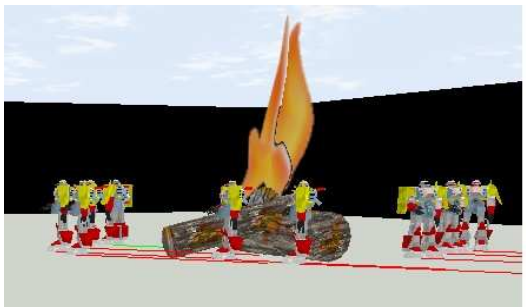

Figure 6: *Team members are circling around a fire location.*

5.2. Oscillate State and Recovery Mode (OSRM) Model

When a robot encounters an obstacle on its way to a target, it may turn to avoid the obstacle then turn back heading toward the target. This implies an agent would turn backward and forward repeatedly. This new model allows the robot to move smoothly alongside the obstacle until it can turn again toward the target. In this model, an agent is in one of two modes. The first mode is when the agent controls and manages its actions according to the LGC model while the second one is when the agent switches to the OSRM mode.

6. A prototype of Fire Extinguishing System

In order to verify the algorithm and the models described in the previous section, an initial prototype has been developed. Our agents were situated in an unknown environment; a virtual laboratory that represents the Leicester Reality Center at De Montfort University shown in figure 7.

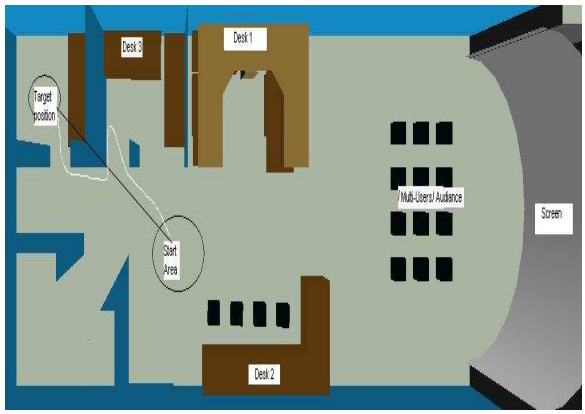

Figure 7: *Virtual Environment Center at De Montfort University.*

A set of agents started the task from the start area (see figure 7) and aim at reaching the fire position and circle around the fire. In figure 7 , the black line shows the shortest path to the fire position, while the white line shows the agents' actual path to the target. To analyze the simulation results and evaluate the system performance, we selected scalability and real time performance as a measure of the system capability. What we mean by scalability is: Is the system limited to a small number of agents or it scales up to a larger number of agents, with the same level of efficiency and safety? In order to investigate the effect of increasing the number of agents on the emergent behaviour, a number of experiments were performed using the different population sizes.

The simulation results,table 1, showed that the system is considered to be scalable for the following reasons: (a) It can handle increasingly complex structures of groups that demand a greater amount of self-organization and motion

Table 1: *Scalability.*

No. of Agents	Frame Rate	Response Time	Completion Time
5	30	0.033	50
10	24	0.42	95
15	15	0.067	150
20	14	0.071	213
25	10	0.1	261
30	9	0.11	298
35	6	0.167	356
40	5	0.2	417
45	4	0.25	465
50	3	0.333	512

coordination without direct change to the underlying mechanisms of the algorithm. (b) All the processing steps and the agents' performance take place in real time (the time interval $\tau = 1/Framerate$ represents the response time). (c) the task completion time linearly increases as the population size increases, figure 8. Figure 9 shows that the response time also

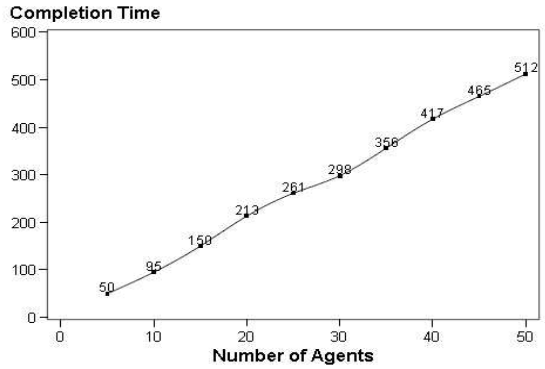

Figure 8: *Task completion time as a function of team size.*

(required time to process the information) as a function of the number of agents. The response time increases as the number of agents in creases.

Figure 10 shows the paths of a set of five agents moving toward the target. Each frame, the positions of the agents and how far they are from the mean of their positions are calculated. Since, the mean of the positions is sensitive to the change in the positions of the agents so any change in the positions of the agents will cause a change in the mean. The graph in figure 10 shows the deviation of the agents positions about the mean. The sum of deviations about the mean, represented by a 2D-delta, equals zero. This is a quite interesting graph it exhibits the flocking behaviour of a set of agents as it shows how agents are able to move away from each other when trying to avoid collisions with each other or encountering an obstacle, presented as a high deviation,

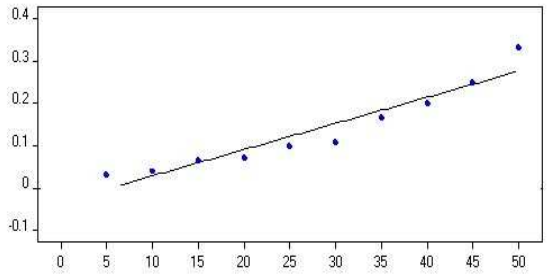

Figure 9: *The response time as a function of population size.*

and then move back again towards each other, presented as a small deviation.

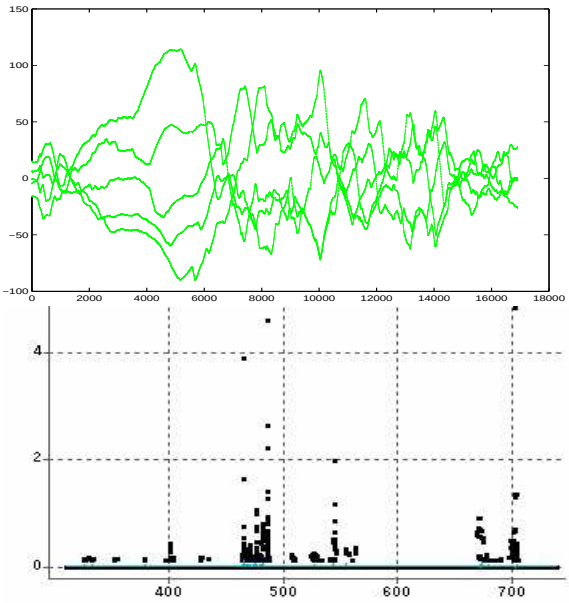

Figure 10: *Top, 2D-delta for a set of 5 agents. Bottom, the cohesion weights along the x-coordinates.*

7. Conclusion

The work presented provides a platform to easily develop different types of behavioral rules and to visualize and interact with agents in a 3D virtual environment in real time. The implemented method focuses on behaviour based motion ,using the flocking algorithms, and blackboard communication techniques. The flocking algorithms as a bi-product aims at minimising extreme clustering of agents, to a known level that again appears to reduce the chance of the worst

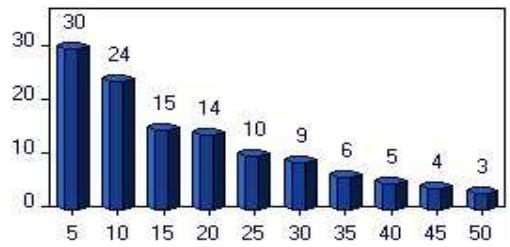

Figure 11: *The frame rate as a function of the population size.*

case scenario. It allows an agent in a large scaled multi-agent system to interact and produce a believable behavior both locally and globally and can be executed at interactive rates.

The use of the Reality Centre allows multi users to assess and evaluate the behaviour during any of the simulation stages and modify the algorithm to suite the application.

The presented work takes account of the real-time considerations for monitoring purposes and timely issuing command-centre actions, but always the internal simulation must take priority with the effect of reducing the frame-rate(figure 11). Also, the incorporation of complex models increases the realism giving extra benefits, but adds complexity to the collision detection algorithms required to see if the sensor detects an object.

References

[AHATI04] AL-HUDHUD G., AYESH A., TURNER M., ISTANCE H.: Agents negotiation & communication within a real time cooperative multi-agent system. *Presented at the Fifth International Conference on Recent Advances in Soft Computing (RASC2004)* (December 2004).

[AHTA04] AL-HUDHUD G., TURNER M., AYESH A.: Speech act and blackboard negotiation based communication protocol for real time multi-agent systems. *Presented at the 2004 UK Workshop on Computational Intelligence (UKCI-04), September* (2004), 112-120.

[Ant03] ANTYCIP UK LTD, DISTRIBUTORS FOR MULTIGEN-PARADIGM, INC. A COMPUTER ASSOCIATES COMPANY: *Vega Prime Training Manual, Version 1.2*, 2003.

[BdL*98] BEER M., D'INVERNO M., LUCK M., JENNINGS N., PREIST C., SCHROEDER M.: Negotiation in multi-agent systems. *The workshop of the UK Special Interest Group on Multi-Agent Systems (UKMAS'98)* (1998).

[WJC03] WATSON N. AND JOHN N. AND CROWTHER W.: Simulation of Unmanned Air Vehicles. In *Proceedings of Theory and Practice of Computer Graphics*(2003), IEEE.

[GM01] GENTILI F., MARTINELLI F.: Optimal paths for robot group formations based on dynamic programming. In *IEEE International Journal of Robotics and Automation* (2001), pp. 197- 206.

[LYS03] LUO C., YANG S., STACEY D.: Real time path planning with deadloock avoidance of multiple cleaning robots. In *IEEE International conference on Robotics and Automation* (September 2003). Tiwan,Thaibi.

[NLJ94] NWANA H., LEE L., JENNINGS N.: *Coordination in software Agent System*. Tech. rep., BT, Queen Marry and Westfeild College, 1994. Intelligent Systems Research Group, Applied Research and Technology Lab, BT Labs.

[Rey87] REYNOLDS C.: Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87* (July 1987), vol. 21, pp. 25- 34.

[TWP03] TANG W., WAN T., PATEL S.: Real-time crowd movement on large scale terrains. In *Theory and Practice of Computer Graphics* (2003), IEEE.

[WCE03] WAN T., CHEN H., EARNSHAW R.: Real time path planning for navigation in unknown environment. In *Theory and Practice of Computer Graphics* (2003), IEEE.