Human Behaviour Modelling in Games using Agents

Susantha Andradi, Asoka Karunananda, and Romesh Ranawana

***Abstract*—** The incorporation of real human behaviour for non-player characters (NPCs) in games is a significant research challenge. The nature of events and actions within a game scenario is associated with temporal and spatial redundancies. Such data cannot be considered in isolation, and should be processed in sequence. Multi Agent Systems are proposed as a potential technology to model natural human behaviour into non player characters in game environments. A three phase solution is proposed for this purpose. During the first phase real data in a game are collected by letting a human player play the game unconditionally and naturally as possible. All objects around the player and each action executed by the player are recorded against the time. The second phase involves the cleaning and normalization of the noise present within data gathered during phase 1. Then for each sequence, data is modelled as HMM along with the player's current action. Rules are generated with these HMM and player actions.

The NPC decision making strategy, consequently are implemented by agents using these rules. HMMs are mapped as resource agents and the NPC in the game is mapped as a request agent. Based on the observation sequence that NPC receives, it broadcasts this information to the resource agents and bids for the best deal and tries to find the most suitable state sequence (that is action sequence). Jade is used as the agent framework and Jahmm is used for HMM modelling. As the test bed (simulation platform) of this experiment, Virtual Battle Space2 (VBS2) along with the VBS2Fusion library is used to prototype the approach. The validity of the proposed method is verified by experiments and analysis by statistical methods. Odds theory is used as one of the statistical methods. Human control missions as well as NPC control missions are recorded as video files. Then arbitrary observers who haven't played this game are used to observe the videos and let them to recognize what is the mission that real player plays and what is the mission that NPC is controlling. According to the results of the observers, it is statistically proven by showing that there is no significant discrimination between real player video and NPC video.

***Keywords*—**HMM, MAS, VBS2, Odds Theory.

I. Introduction

THE primary purpose of gaming technology as an entertainment medium is rapidly changing with the advent of game based technology for education and training. For example, military training requirements are increasingly fulfilled by strategic multiplayer or single player games [7]. Real soldiers are assigned as the players of the game scenario and are allowed to interact with computer controlled characters, also known as non-player characters (NPCs), who can simulate either friend or foe. It is therefore vital for the realism and effectiveness of the exercise for the NPCs to exhibit the realistic human behaviour and to also adapt to situations based on environmental factors, skill and priorities. Gaining true realism has uniformly been a major obstacle in game design. This research area has therefore become a hot topic as there are still many issues which are yet to be addressed in order to create natural and realistic NPCs.

The common method of incorporating fair intelligence to the NPCs is the addition of a rule based Finite State Machine (FSM) or decision tree [6]. But the problem of the predefined rule set is the inability to adapt to a rapidly changing dynamic environment. Hence it is not enough to create a realistic dynamic environment. The user may find the game boring when he realizes that the game AI responds the same way when the environmental factors are very much similar [1]. Strategies and tactics of NPCs may not dynamically change when the decision trees are pre-stored or generated [2].

In the early days researches mainly focused on FSM based experiments and it has given sufficient results compared to the technological advancements at that time. It was popular due to the simple mechanism used to write the rules and the accuracy [8]. Due to the complexities of the games, FSM is no longer considered as the best approach for implementation. Nowadays researchers are working on different areas to come up with better intelligence for the NPC behaviours.

Goal oriented meta rules were written and they focused on the target achievements of the game [9]. According to Shi et al, meta rules are the fundamental design elements of the game, the AI is automated by considering all possible states to achieve meta rules and backpropogate them using ANN. Since ANN considers all possible states, it is not recommended for use in the network multiplayer game, instead they propose to use this for a single player game. Another research is the combination of ANN and evolutionary computing done by Cho Sung-Bae et al [10]. Initial data are formed by considering a small number of rules and then they perform genetic operations and the initial data is considered as the data pool. A similar kind of research carried out by Frank Dignum and Joost Westra [11], is an evolutionary neural network for

Susantha Andradi is with the Simcentric Technologies and a student of Faculty of Information Technology, University of Moratuwa, Sri Lanka, (corresponding author's phone: +94772200958; e-mail: susantha.andradi@gmail.com).

Asoka Karunananda is with the Faculty of Information Technology, University of Moratuwa, Sri Lanka(e-mail:asoka@itfac.mrt.ac.lk).

Romesh Ranawana is with Simcentric Technologies. (e-mail: romesh@simct.com).

http://dx.doi.org/10.15242/IIE.E0314005

Quake III. They have mainly focused on item selection and weapon selection. Before feeding the input values, they have normalized them to range between 0 – 1. Another way of modelling human behaviour is to make a self-organizing neural network [12]. Here self-organizing means, not necessarily to design number of nodes, layers in the network.

In order to improve NPC behaviour, Baysian based probabilistic mechanism has been used for making decision model [13]. A Bayesian network is used as a tool for gathering all experience data (during the game) and Fuzzy model is constructed to define certain situations. So the programmer can improve the NPC behaviour with the least amount of effort. Similar kind of method; probabilistic mechanism was used to solve the dynamic game difficulty adjustment issue [14]. They propose to gather player information while he is playing the game and evaluate his abilities. During the game, NPC behaviours are refined so that the difficulty of the game is automatically dynamically adjusted.

All the states and actions can be treated as sequential, partial data. So Hidden Markov Model has been applied by Hao Wang for reinforcement learning for NPC behaviour modelling [15]. There are two types of NPCs, one is the commander and the other is a soldier. Commander generates the goals (through reinforcement learn – Q-learning-style algorithm) and send goals as orders to the soldiers. Then the soldiers behave as goal oriented agents (based on a hierarchical task network). According to the UT architecture, AI can be integrated through gamebots. Consider State-Action pair and apply them during the game. They use Q-learning as an algorithm to solve the Markov Decision Process. Another interesting method of human behaviour modelling with multi-dimensional time series has been considered here [3], [4], [5]. First they record all possible information around the human player during the game. Then by applying certain data cleaning techniques, this information is cleaned and subsequently clustered. Finally HMM is applied to these data to consider NPC decisions.

The authors propose a method for automated decision making system generation which utilizes the perceptions and actions gathered through human players to determine probable action for NPCs. The primary goal is to mine perceptions and related actions of real players to incorporate within NPCs decision making structures. Dynamic action decisions are made, based on the stored data through agent communication.

This report is structured as follows. Section II and III explain the novel approach using agent technology for realistic NPC behaviour. Section IV is devoted to explaining the implementation of the proposed system. Experiments are presented in section V and finally the conclusion of this proposed approach and further work are described in section VI.

In this section some background information about the problem domain and a little bit about the solution that we adapted are discussed. The weight of the problem is given, so the convincing solution found is highlighted. In the next section main focus is to describe the proposed approach.

II. Agent Based Dynamic Human Behaviour Modelling

In the real world, most of the examples are considered as continuous HMM. This probabilistic mechanism can be used for evaluation, training or recognition of many problems. Though the same rules are used, through the agent communication, dynamic behaviours can be generated because of the features of MAS. In this section, a brief discussion about the proposed approach is carried out and how these technologies are used in order to solve the problem is discussed.

The hypothesis of this approach is that although the same data are used, behaviours of NPCs can be dynamically applied using agent communications. This idea is inspired by the way how different game players like cricketers are playing with different strategies even though the cricket rules are same for everyone. A three phase solution is proposed in which the first phase is to store real human behaviour, the second phase is to clean them and the third is to use them during a real game for making decisions. As agent techniques give promising results in complex situations, the same techniques can be used to control NPCs in the games because of game nature is partially observable, stochastic, non-deterministic, dynamic etc. Human behaviour modelling with multi-dimensional time series is an inspiration for us to use the similar technique for all three phases [3], [4], [5]. But the novel approach is agents taking the responsibility of making decisions by considering the HMM results. So the strategies of each agent would change even though the same rules (HMM models) are considered.

Letting the real human players play the game as naturally as possible is the very first step towards recording necessary information. Then these data are used to model many HMM models in different occasions. Each generated HMM model is assigned to resource agents respectively. NPC, which is considered as the request agent broadcasts its observable patterns in each time step and if the likelihood probability of an HMM would match, it is considered as the best deal and NPC applies the relevant action of the HMM.

Our hypothesis is though the same data are used, and the same rules are generated, behaviours of NPC can be dynamically generated through the agent communication. As described above, the approach consists of three main phases which are data recording, data cleaning and processing to generate rules, and to make decisions through agent communication. In the section IV the design of the proposed solution is described in terms of each main phase.

III. Design of Dynamic Human Behaviour Agent

In the previous section, the approach of the human behaviour modelling was described. Three phase mechanism is to first record the data during the real game play, second clean the data and generate models as rules, finally agents who represent both data elements and NPC to communicate with each other and determine the most suitable action or action sequence for a given observation sequence during the game. In this section approach is described in detail by elaborating the design components of the proposed system. The three main phases of data recording, data cleaning and apply the behaviours are as follows.

A. Data Recording

In order to decrease the probability of the deterministic behaviour of the game, various human behaviour models are added. The approach was to let different humans to play the game and record their behaviours. The idea of phase one is, it can be used to host a scenario where multiple players can join with the mission and let them play without any conditions.

So players must play naturally as possible, while they are playing, some specific data are recorded. Player's current direction is considered as the main direction relative to the map north, for states categorization (Fig. 1). A radius of the 8m circular area around the player is considered for determining observation patterns. Any object that resides within this area is recorded at each time step of the game based on the direction relative to the player, they are discretely considered as observation status. 18 numbers of discrete observation states are defined. For this purpose, 12 zones around the player are considered (Fig. 2). Based on the locations of the other objects around player, observation state is determined. Some of the observation states are illustrated in Fig. 3, Fig. 4, Fig. 5, Fig. 6 and Fig. 7. As the discrete states can't cover all the possible observations, 'Ignore' status is there to represent all the undefined observations.

Fig. 1 Possible Player Statuses (Directions)

Apart from these objects around the player, the actions that player made were recorded. Player's direction is considered as the player action, so the assumptions are that the player has constant speed and objects that reside at more than 8m in circular area are none of its concern. Player actions (changing the direction) depend on the observation patterns he seeks.

B. Cleaning and Clustering

This is the second phase of the approach. Recorded data may be noisy and redundant. They need to be cleaned first. For example, in human running, when the player runs for certain event if he is on the road, he may use the pavement or he may use the road itself if no vehicles are around. So when these movement patterns are considered, these will have slight differences, but they can be considered as similar pattern of movement. During phase two this kind of slight different movement patterns is considered into one cluster and form a general pattern for the given player records. This general pattern of data is considered by Hidden Markov process. In this proposed method, Continuous Dynamic Programming (CDP) [16] approach is used to clean the data. Each pattern of data is evaluated using CDP and data are modified consequently input to generate the Hidden Markov Models (HMMs). Once the HMMs are generated they are stored against current action and the adjacent action took place during the game. Then a set of rules is generated as Equation (1).

If {Current Action && HMM(Tn)} Then { Next Action} (1)

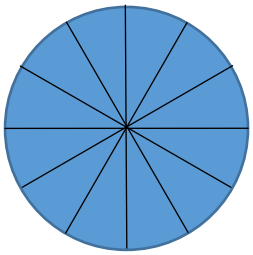

Fig. 2 Areas to be considered for observations

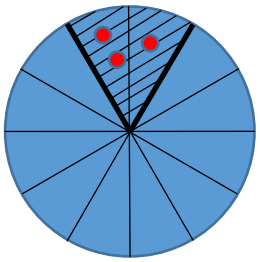

Fig. 3 Critical Status

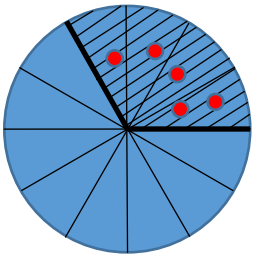

Fig. 4 Right Covered Status

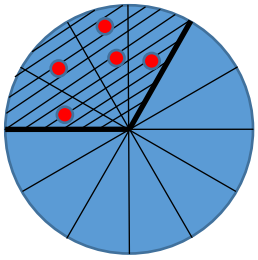

Fig. 5 Left Covered Status

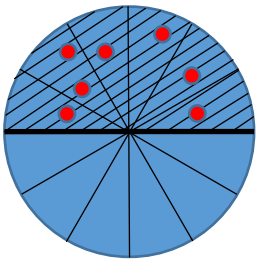

Fig. 6 Covered Status

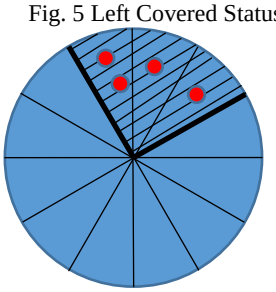

Fig. 7 Right Critical Status

C. Applying the Behaviours

Each NPC in the game is mapped by a request agent. Each request agent has access to its own data model which is created during phase 2. If there are 10 different NPCs available, 10 different data models which are created using 10 different players are used. (For large scale NPCs like crowd ambience, data models are redundant among a group of NPCs). Though particular agent has a particular data model (like his mind and the experience), it communicates with other request agents in the scenario who have different data models and take the rational action.

Rather than consider the same set of rules in a static manner, agents execute rules dynamically changing the strategies. Each agent uses the data driven by HMM, as game data are spatial

and temporal [3], [4], [5]. Output is the action which is to be applied to the relevant NPC in the game. High level architecture of phase 3 is shown in the Fig. 8.

Virtual Battle Space2 (VBS2) as the simulation platform on the Windows7 operating system is used. VBS2Fusion is the library which can be used as a tool to access VBS game engine in order to do the simulation. Jahmm (pronounced as Jam) is used to model the data patterns using Hidden Markov. Jade with the Java development is used to emulate the agent framework.

In order to prove the hypothesis is correct, the test of this solution is very important. First by letting human players to play the game, record their states and the results. Then the same game scenario can be used for letting NPCs play and take the states and the results. Then compare both cases and check the accuracy of the solution.

IV. Implementation

Discussion on the design of the previous section highlighted the main components of the design. Making the data warehouse is the first step towards modelling the HMM. As the second step, these data in the warehouse are processed and the rules are generated as equation (1). Then through the communication, best deal resource agent is considered by the request agent (NPC in the game). In this chapter we are going to have a look at the implementation of each of these design components.

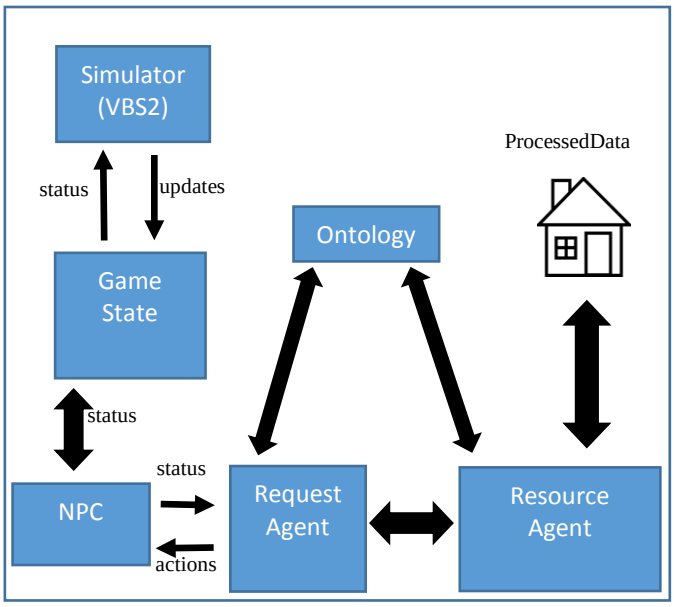

Fig. 8 Top Level Architecture for the phase 3

A. Main Implementation

Main implemented components are shown in Fig. 9.

1. Data Recording

A data recording option is given through VBS user action. When the commands are given, the C++ program (plugin dll) starts the recording of information around the player in each simulation step. The communication between simulation platform (VBS) and the C++ program is done by VBS2Fusion API. As VBS2Fusion is a C++ library, C++ plugin could record each observation data and state data (user actions) in each time step (Fig. 10). The representation of each data segment is like in Fig. 11. It contains both player information (direction, speed) and observation information (other objects' positions around the player, their speeds and directions etc.). A sequence of data segments is generated during player's one particular action. So that during one game mission, multiple sequences of data segments are created and stored in a list.

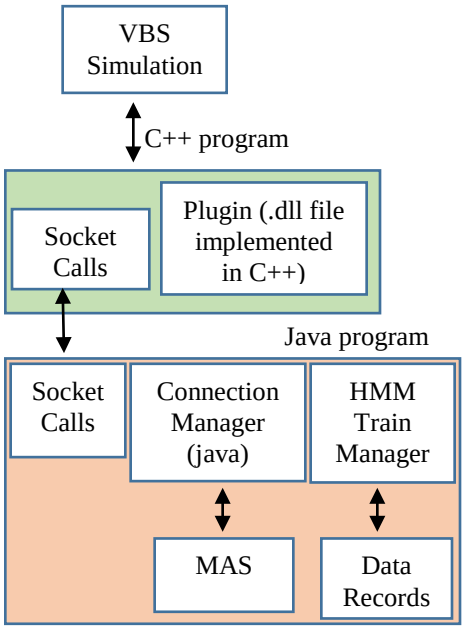

Fig. 9: System architecture

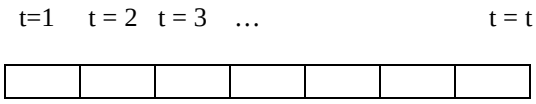

Fig. 10 Sequence of data segments

2. Data Processing

The generated list of data segment sequences is processed after the game finishes its data record. This implementation is also done in C++ program. During this phase the transition probability matrix and the emitting probability matrix for each sequence of data segments are calculated. For each sequence of data segments, separate external files (ProbabilityMatricesx.dat) are created. The naming convention of this file is as stated in the previous line. x denotes the number of data segment sequences. (Ex. ProbabilityMatrices1.dat, ProbabilityMatrices2.dat, etc.).

Once the probability matrix files are ready, the NPC controlling session could be started. Then when the Java program is run, these files are processed by the Java program which uses the Jahmm Java library for the Hidden Markov Model generation using each probability matrix. Initial probabilities are dynamically calculated by considering each

probability matrix, because the HMM are generated by considering the particular Markov state per model. Generated HMMs are stored in a list along with the user's current action and the next action.

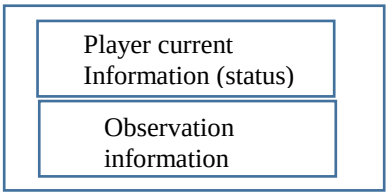

Fig. 11 Representation of the data segment

3. Apply Behaviours Through Communication

This system is designed in a way, so that the Java program is the server and C++ program is the client for socket communication. So the Java program must be run first, before C++ program is run. As a command line argument, the number of required resource agents is defined. This number of agents are created during the start of the Java program. Each resource agent at the time they are created is assigned for each probability matrix file so that they could calculate their HMM. When the MAS is started, each resource agent knows their HMM and the request agent who represents the NPC, negotiates with the resource agent to get the best deal according to the observation information in each time step.

Multi Agent System is populated using the Jade agent framework. Then during the real game where the real human behaviours are applied for NPCs, C++ plugin and Java communication are done through Java and C++ socket communication.

NPC is mapped as a request agent and it sends regular observation updates as a sequence of data to the Java application then through the communication of MAS, most suitable and likelihood HMM model is picked (Model consists of HMM model, current action and next action). If the current action of the NPC and the current action of an HMM model are matched, then the next action of an HMM model is applied to the NPC. But the cognitive part of the NPC comes when it decides that the resulted next action from the proposed rule is not suitable according to the observations. Request agent can change the proposed action if it is not suitable for the NPC.

In this section main components of the implementation are discussed. C++ plugin is developed for data capture while the real human player plays the game and the resulted probability matrices were recorded to external files. These files are considered by the Java program which is executed by using the Jahmm library for making HMMs. HMMs are stored along with the player current action and the next action. The last phase is to initiate the agent framework. The Jade Java library is used to emulate the agent platforms. HMMs along with player's current action and the next action is considered as a resource agent. NPC in the game is considered as the request agent. Communication between C++ and Java is done through socket communication using Xps library. The next section focus is on the evaluation.

V. Evaluation

The performance of the NPC behaviours is tested based on two factors. The main feature of NPC must be hiding the fact that it is artificial. Their behaviours must cheat the observer to recognize them as NPCs. The other factor is the time taken by NPCs for achieving the target. For these purposes, 12 real player missions and 12 NPC missions are used.

A. Human like NPC Behaviours

This is done with a questionnaire given to a human observer. He is given few videos which were captured while a real player was playing the game. Among them there were few videos which NPC was playing the game. Results were obtained based on the how observer could identify which one was the real player and which one was the NPC. The results of the observer can be categorized as follows.

- Observer recognized the game was played by NPC and truly NPC was playing. (NN)
- Observer recognized the game was played by NPC and truly real player was playing the game. (NP)
- Observer recognizes the game was played by real players and truly NPC was playing the game. (PN)
- Observer recognizes the game was played by real players and truly real player was playing the game. (PP)

These categories can be used to measure performance of the human behaviour model statistically.

TABLE I
Possible Results of the Observer

		True state of video file	
		NPC was playing	**Real player was playing**
Predict state of video file	**NPC was playing**	6	6
	Real player was playing	7	5

For a better performance, the ratio between the correct classification and the wrong classification should be low. Odds theory can be considered to measure these statistically.

Value for odds ratio can be in between 0 to infinity. The value 1 means there is no significant discrimination between real player video and NPC video. Therefore, the 95% confidence interval for the log value of the odds ratio can be calculated using "(2)". The Standard error can be calculated using "(3)". Calculating antilog provides the confidence interval for these values. If the confidence interval includes 1, that means there is no significant discrimination between real player video and NPC video.

$$odds\ ratio = \frac{PP \times NN}{PN \times NP}$$

According to the Table I, the calculated odds ratio was 0.71428 and the confidence interval for the odds ratio was {0.205, 3.037}. The calculation was given in "(4)", "(5)", "(6)", "(7)", "(8)", "(9)", "(10)" and "(11)".

$$C.I = \log(OR) \pm 1.96 \times SE(\log OR) \quad (2)$$

$$SE(\log OR) = \sqrt{\frac{1}{NN} + \frac{1}{NP} + \frac{1}{PN} + \frac{1}{PP}} \quad (3)$$

$$odds\ ratio = \frac{6\times5}{7\times6} \quad (4)$$

$$odds\ ratio = 0.71428 \quad (5)$$

$$\log Odds\ ratio = -0.1461 \quad (6)$$

$$SE(\log OR) = \sqrt{\frac{1}{6} + \frac{1}{6} + \frac{1}{7} + \frac{1}{5}} \quad (7)$$

$$SE(\log OR) = 0.6761 \quad (8)$$

$$C.I = -0{:}1461 \pm 1.96 \times 0.6761 \quad (9)$$

$$C.I = -1.4712, 1.1790 \quad (10)$$

This is the confidence interval for the log (OR). In order to calculate the confidence interval for odds ratio, the intervals should be converted into its antilog. Equation 11 gives the antilog of the limit.

$$C.I = 0.0779, 15.1 \quad (11)$$

This confidence interval includes 1. Thus, it can be concluded that the odds ratio is approximately equal to 1. This shows that there is no significant difference between correct identification and wrong identification. Thus, it can be assumed that the selection was done randomly. Therefore, the behaviours of the NPC are more similar to the real player behaviours.

B. The Time Spent for Achieving the Target

Average time taken for achieving the target by real player was compared with the time taken to achieve the target by NPC.

TABLE III
AVERAGE RESULTS OF THE TIME

Human Model	Time taken by real human (Average in seconds)	Time taken by NPC (Average in seconds)
Human 1	28	32
Human 2	35	38
Human 3	29	32
Human 4	36	34
Human 5	43	38
Human 6	42	40
Human 7	37	35
Human 8	34	39
Human 9	48	42
Human 10	29	35

According to the experiment results in table II it shows that the NPCs trained by this system behave somewhat similar to how the real human players do.

VI. CONCLUSION

The main purpose of this research work is to show MAS and HMM can be used to model realistic human behaviours in games. Prototype of the proposed approach is implemented as a Java application with c++ plugin using some external libraries. Simulation is modelled in VBS2. The experiment is carried out in two criteria to prove them statistically. First one is to show the NPC behaviours are human alike. Recorded videos while playing the game by real player and NPC were mixed each other and they were given to an unbiased observer. Let observer to classify the videos by checking the actions of the game entities to recognise what is real and what is artificial. Odds theory is used to prove that there is no significant discrimination between real player actions and NPC actions. The second criterion to prove the obtained results statistically is to check the time that NPC has taken to achieve the target while comparing the time that real player has taken. It is proved that there is no considerable difference between NPC accuracy and real player accuracy. Hence the conclusion of this research is human behaviour can be modelled realistically using agent technology and hidden Markov approach.

REFERENCES

[1] X. Chen, Y. Gao, H. Wang, RL-DOT: A reinforcement learning NPC team for playing domination games , IEEE Transactions on computations on computational intelligence and AI in games, Vol. 2, No. 1, 2010.

[2] P. Cunningham, M. Fagan, C. Fairclough, B.M. Namee, research directions for AI in computer games, Trinity College Dublin, Department of Computer Science, TCD-CS-2001-29, 2001, pp 5-12

[3] K. Doki, K. Hashimoto, S. Doki, S. Okuma, A. Torii, Estimation of next human behavior and its timing for human behavior support, International Conference on Control-Automation-Robotics and Vision, 2010, pp.952-957.

[4] K. Doki, K. Hashimoto, S. Doki, S. Okuma, A. Torii, Modeling and recognition method of human behaviors with multi-dimensional time series data, Systems Man and Cybernetics (SMC), 2010, pp. 2058-2063.

[5] K. Hashimoto, K. Doki, S. Doki, S. Okuma, (2009), Study on modeling and recognition of human behaviors by If-Then-Rules with HMM, Industrial Electronics IECON '09, pp. 3410-3415.

[6] C. Lee, C. Park, J. Kim, S. Youk, K. Ryu, An Intelligent NPC framework for context awareness in MMORPG, International Conference on Convergence and Hybrid Information Technology, 2008.

[7] M. Mozgovoy, P.C. Rogers, I. Umarov, Believable and effective AI agents in virtual worlds, International Journal of Gaming and Computer Mediated Simulations, Vol.4 Issue 2, 2012, pp.37-59.

[8] E. Long , Enhanced NPC behavior using goal oriented action planning, University of Abertay Dundee, 2007, pp 14-16.

[9] I. Umarov, M. Mozgovoy, C. Rogers, believable and effective AI agents in virtual worlds : Current State and Future Perspectives, International Journal of Gaming and Computer-Mediated Simulations, Vol. 4(2), 2012, pp. 37-59.
http://dx.doi.org/10.4018/jgcms.2012040103

[10] S. Cho, S. Jang, J. Yoon, Optimal strategy selection of non-player character on real time strategy game using a speciated evolutionary algorithm, CIG'09 Proceedings of the 5th international conference on Computational Intelligence and Games, 2009, pp 75-79.

[11] F. Dignum, J. Westra, Evolutionary neural networks for non-player characters in Quake III , IEEE Symposium on Computational Intelligence and Games, 2009, pp.302-309.

[12] S. Feng, A. Tan, Self-organizing neural networks for behavior modeling in games, International Joint Conference on Neural Networks, 2010, pp.1-8.

[13] Y. Hong, Z. Liu ,A preliminary research on decision model based on bayesian techniques for a NPC in computer games, International Symposium on Computational Intelligence and Design, 2010.

[14] V. Chapman, Hunicke, Robin, AI for dynamic difficulty adjustment in games, International conference on advances in computer entertainment technology, 2003, pp 429-433, ISBN.1-59593-110-4, Northwestern University.

[15] W. White, A. Demers, C. Koch, J. Gehrke, R.Rajagopalan, Scaling Games to Epic Proportions, Cornell University, 2007.

[16] O. Jet et al, Motion Recognition using DP Matching, Journal of the Japan Society of Precision Engineering,Vol.63, o.6, pp.812-818,1997