Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 13 (5): 683-691, 2013
ISSN 1812-5654 / DOI: 10.3923/jas.2013.683.691

A Multi-agents Coordination Mechanism to Improving Real-time Strategy on Tower Defense Game

[1]Steven K.C. Lo, [2]Huan-Chao Keh and [2]Chia-Ming Chang
[1]Department of Information Management, Jinwen University of Science and Technology, New Taipei City 231, Taiwan, Republic of China
[2]Department of CSIE, Tamkang University, New Taipei City 251, Taiwan, Republic of China

Abstract: This article uses the multi-agent system coordination mechanism and implements the characteristics of agents to construct and add intelligence to a computer program. By redesigning and developing a tower defense game as an example, this article presents models for coordination and message transmission between agents and creates two new methods to correct the path of the enemies and dynamically generate agents which largely improves the intelligence of the enemies and towers in the game. Overall, the system changed traditional game rules and made a real-time strategy game more challenging.

Key words: Multi-agent, A*, pathfinding, intelligent agent, tower defense

INTRODUCTION

An intelligent agent (Bonabeau, 2002; Kinny and Georgeff, 1997; Rybakov, 2009; Wooldridge, 1997) is a computer program that acts as background service to observe the environment and gather information. An intelligent agent uses the parameters provided by the user and works independently to achieve a set of predefined goals. Typically, an agent has the characteristics of autonomy, social ability and reactivity. An agent with the autonomy characteristic can employ knowledge and make decisions to complete a given task automatically. Social ability allows an agent to exchange information with other agents. A Multi-Agent System (MAS) (Balaji and Srinivasan, 2010; Chen *et al*., 2011; Ferber, 1999; Horling *et al*., 2006; He *et al*., 2009; Ilarri *et al*., 2008; Kotz and Gray, 1999; Lo, 2012; Park and Sugumaran, 2005; Shah *et al*., 2009; Tornero *et al*, 2012; Wu, 2001; Yau *et al*., 2003; Zambonelli *et al*., 2003) consists of intelligent agents that cooperate and coordinate with one another to accomplish tasks. The system can be used to solve problems that cannot be solved by individual agents.

Tower Defense (TD) (Avery *et al*., 2011) is a type of Real-Time Strategy (RTS) game, where the player defends a castle against waves of enemy attacks. Every time the player defeats an enemy, the player gains resources as reward which can be used to build different types of defense weapons or upgrade existing weapons to make them more powerful. In the traditional TD game, the path of the enemies is predefined for each map. All of the enemies will move along the path without searching for the destination. Although, using predefined paths greatly improves performance, it limits the challenge of the game because players can clearly predict the path. The shortest path algorithm (Cherkassky *et al*., 1996; Gallo and Pallottino, 1998; Zhan, 1997), a common AI design in videogames, finds the shortest route between two points. There are several pathfinding algorithms and the most common used in videogames is A* (Goldberg and Harrelson, 2004; Goldberg *et al*., 2006) which is derived from Dijkstra. The advantages of A* is that it calculates heuristics to exclude the wrong waypoint and that it performs better than Dijkstra (Yershov and LaValle, 2011).

In multi-agent system studies, research articles focus on the industry application, environment development, principles, representation and others areas, as shown in Table 1. Although, the multi-agent system is rarely applied in the A* field, multi-agent coordination mechanisms are used to effectively handle real-time strategy in a tower defense game.

This article presents two new methods: (1) A path correction method which uses A* as our basis and applies the autonomy characteristic of the intelligent agent to replace the traditional pathfinding solution of the TD game so that enemies no longer move along a fixed path and can adjust their waypoint to prevent attacks from towers and (2) A dynamic enemy generation method which will change the enemy type and the number of enemies generated in each wave, depending on the current power of the player, based on the difficulty level that the user selected, as well as other game variables.

Corresponding Author: Steven K.C. Lo, Department of Information Management, Jinwen University of Science and Technology, New Taipei City 231, Taiwan, Republic of China

Table 1: Comparison of similar research

Article	Agent based	A* field	Solving problem	Solving method
*MACM	Yes (multi agent)	Yes	Improvement of path correction and dynamic enemy generation	Multi-agent coordination model
Kotz and Gray (1999)	Yes	No	Hosting and willingness to host some form of mobile code or mobile agents	Mobile-agent technology
Shah *et al.* (2009)	Yes (multi agent)	No	Lack of uniform representation of exceptions in open multi-agent systems	Ontological approach
Ilarri *et al.* (2008)	Yes (multi agent)	No	Continuous monitoring of highly dynamic distributed environments	Divide-and-conquer cooperation structure
Yau *et al.* (2003)	Yes (multi agent)	No	Overcame the low bandwidth and poor network connectivity problem	Multi-resolution transmission and browsing mechanism on Distributed Agent Environment (DAE)
Goldberg *et al.* (2006)	No	Yes	Improvement of the reach-based approach of Gutman (Gutman, 2004) in several ways	Point-to-point shortest path
Liu and Yao (2004)	Yes (multi agent)	No	Methods used by the agents to compete and cooperate in a UAC to provide services	Ubiquitous Agent Architecture (descriptive entities, functional modules, messages and protocol)
Goldberg and Harrelson (2004)	No	Yes	Use of the A* search to improve the shortest path algorithm	Graph-theoretic lower-bounding Technique based on landmarks and the triangle inequality
Ozen *et al.* (2004)	No	No	System scalability when the number of users increases dramatically and methods for expressing highly personalized information in the user profiles	XML and finite state machine (FSM)
Tweedale *et al.* (2007)	Yes (multi agent)	No	Methods for interacting with humans on Distributed Artificial Intelligence (DAI)	JACK framework
Zambonelli *et al.* (2003)	Yes (multi agent)	No	Software engineering applications in complex domains	Gaia methodology
Lin and Tsai (2009)	No	No	Transportation problem with fuzzy demands and fuzzy supplies	Genetic algorithm (GA)
Chen *et al.* (2008)	No	No	Transportation costs for routes	Fuzzy Network

*MACM: Multi-agent coordination model

SYSTEM STRUCTURE

This article uses some basic game components provided by Unity to develop a tower defense game. The system architecture is shown in Fig. 1:

- **User interface:** The UI consists of scripts and textures which will handle different instructions from the user
- **Game controller:** The core of the game which is responsible for the workflow of the game and states, such as checking win or lose conditions and it also receive commands from the UI and delivers them to the right game object manager
- **Game object management:** There are several game object managers, including towers, enemies and waypoints; these managers are responsible for the management of their subagents
- **Terrain:** A plane consists of triangles that are positioned as ground in the game
- **Physics:** Gravity and collisions keep objects on the ground and allow them to collide with each other
- **Particle system:** Creates particles to simulate effects, including fire, explosions, smoke and sparks
- **GUI:** Provides basic graphic user interface components, such as panels, buttons and interactions
- **Audio:** Plays the background music and sound effects

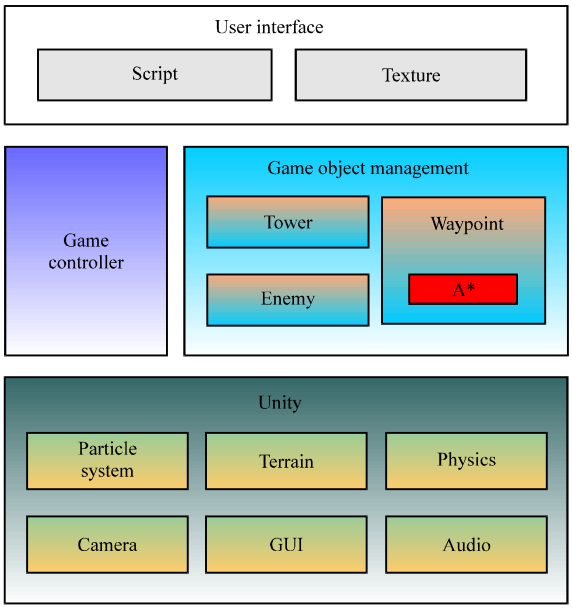

Fig. 1: System diagram

- **Camera:** Captures and display the game view to the player

In the general design of the TD game, the towers that are built by the player oppose the enemies. The player can build and upgrade different types of towers to defeat the

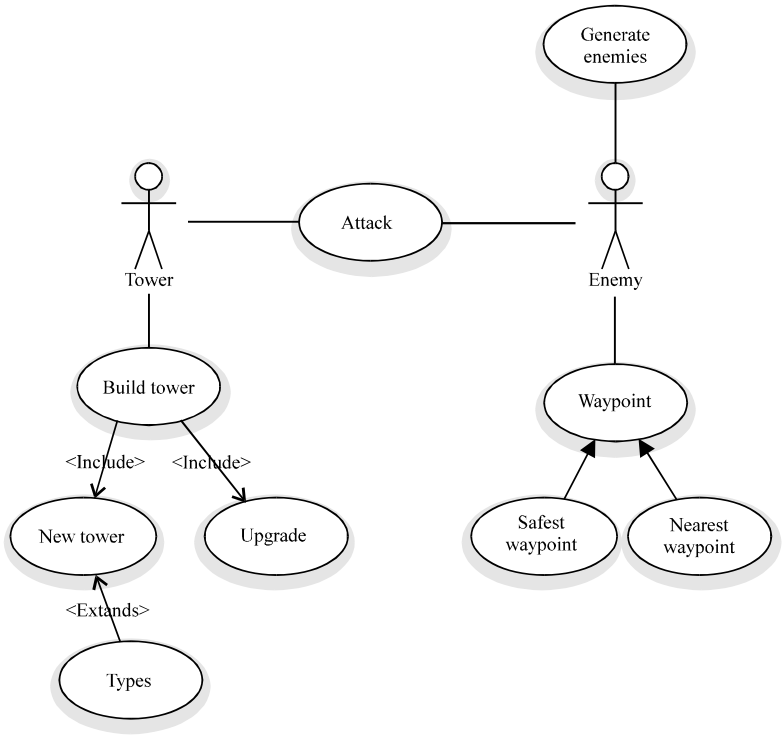

Fig. 2: Use-case diagram of tower and enemy

enemies. Every tower has an attack range and will automatically attack any enemies in range of the tower. In contrast, the enemy is just following the path predefined in different maps which is the most important part of a TD game because it affects how the user considers tactics and how enemies react to the player actions. The diagram of this process is shown in Fig. 2.

MULTI-AGENTS COORDINATION MODEL

Here, focuses on the features of multi-agent interactions and discusses the existence patterns of their interactions. When an agent wants to communicate with other agents, it cannot go beyond the scope of its basic mode of communication. There are many different types of communication among agents, such as one-to-one, one-to-many and many-to-many interactions. This article is based on an Agent Announcement State Transition Diagram which represents an agent's intention to communicate or interact with another agent. If an agent wants to access information from other agents or wants to share its own information, it must declare this intention. The Announcement State Transition is shown in Fig. 3.

In Fig. 3, there are five possible states: Announcing, Suspending, Mutating, Critical and Accepting. These states represent the possible states of the announcement process if one agent wants to communicate with another agent. Following are explanations of the significance of each state:

- **Announcing:** The initial state of an agent. This state indicates that this agent can communicate with other agents and pass messages
- **Suspending:** When the agent completes its task or is no longer able to coordinate, it can return from the Suspending state to the initial Announcing state to wait
- **Mutating:** After the implementation state of the agent, if there is any other declaration of implementation, the implementation can return the state of the agent to Announcing through a Mutating state and re-coordination
- **Critical:** The agent enters this critical state through a sequential process of coordination and implementation
- **Accepting:** When the agent can only accept messages from other agents, it enters this state

Here, present the coordination model (Ito *et al*., 2002; Robertson, 2004; Sycara and Zeng, 1996) of different agents. All game objects will be redesigned as agents to

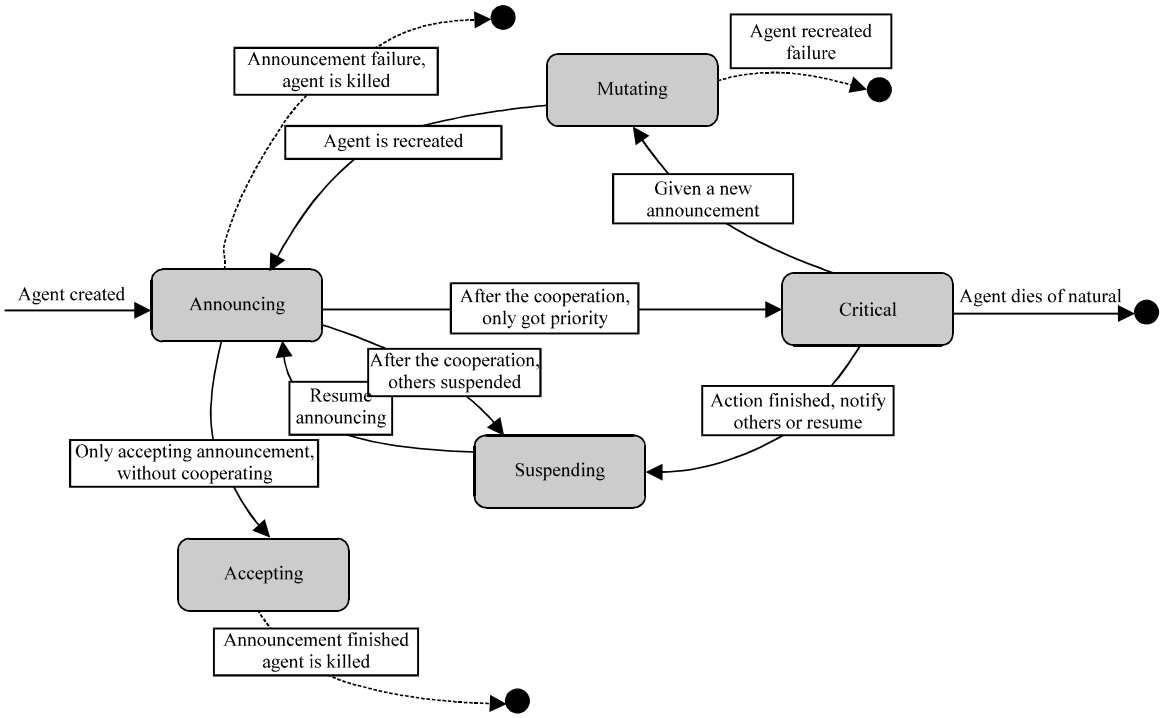

Fig. 3: Agent announcement state

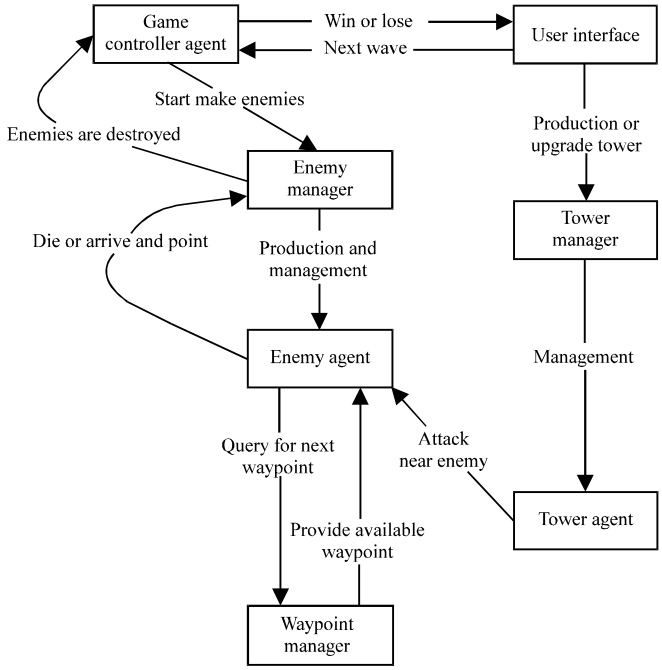

Fig. 4: Overall coordination model of agents

make them smarter and apply the characteristics of intelligent agents. Agents should be able to gather environmental information in every game round and make the right decision according to that information or past experiences. The overall coordination model of agents is shown in Fig. 4.

Manager agent: Managers are represented as agents and are mainly responsible for management, including creating and deleting subagents. Managers also receive all the statuses reported from subagents. When a subagent is created, the manager will sort the subagent into a group that contains agents of the same type. Agents are able to communicate with other agents in the same group to exchange information. In addition, the enemy manager will also evaluate the power value of the player which consists of several types of data, such as how fast enemies are to be killed in a round, the round counts and the average damage of the tower that hits the enemies. Those data come from the enemy manager's subagent report in every round.

Tower and enemy agent: The enemies' goal is to cross the map and arrive at the destination of the waypoint without being killed. To achieve this goal, each of the generated enemies has to gather as much information as possible about the environment, including the nearest waypoint which comes from the waypoint manager, or the tower position which is reported from other agents. If any agent receives enough information to determine the best waypoint, it will communicate with other agents in the same group and share the information. The movement path will be reported to the enemy manager, as shown in Fig. 5, to decrease the computing time of other agents and improve overall performance. If any enemy is killed at the half-way point, every agent in the same group will also receive the notification.

Towers are different from enemies. The player chooses the placement of the tower. However, the attack mode can be selected as manual or automatic. The player can select their target manually in the manual mode. In automatic mode, the tower will simply attack any enemies in the attack range. According to the game rules, the player should prevent enemies from arriving at the end point. Therefore, the tower needs to be smart and prioritize killing the weakest enemy. The difference in the behavior of the two modes is shown in Fig. 6.

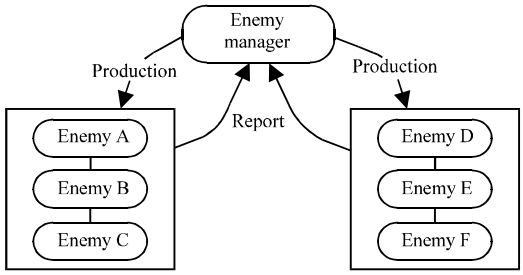

Fig. 5: Message transmission between manager and agents

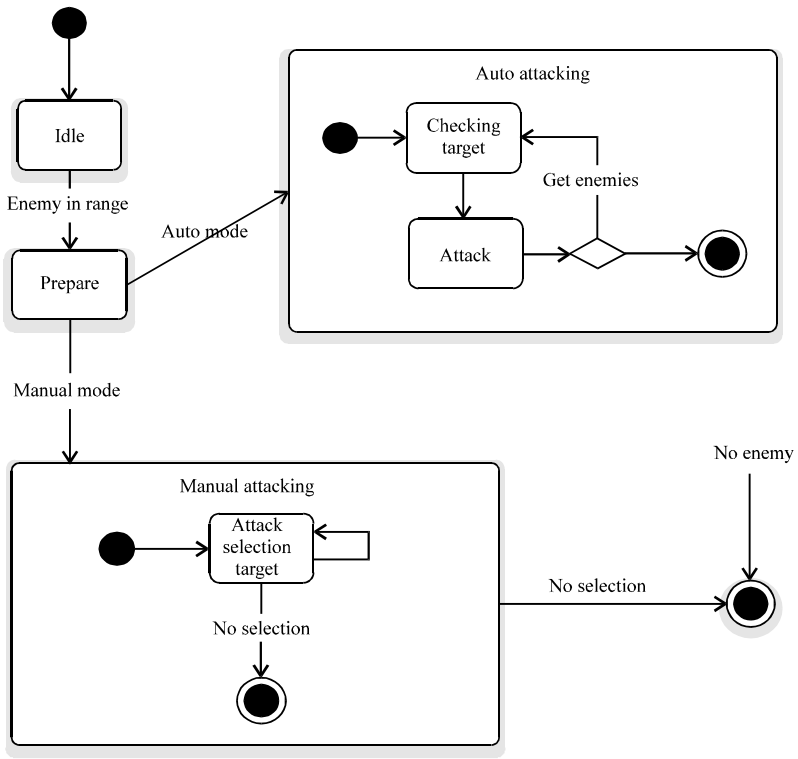

Fig. 6: State diagram of tower attack modes

Waypoint agent: Every waypoint has a weight value, with a higher value meaning that going to the waypoint is safer and a lower value meaning the opposite. Weight is affect by a few symptoms, including the decrease in the amount of hit points of enemies and the amount of enemies destroyed. Every few rounds, the waypoint manager gathers more information from various sources. These stored data will help agents decide their next movement path which prevents an enemy from easily being destroyed, as shown in Fig. 7. Once the walking path of enemies can be changed dynamically, the player must consider the placement of towers more carefully in each round to defend against the enemies. However, if the path changes frequently, the player may never be able to catch any enemies. Therefore, the agent will only compute the new path at the end of each round.

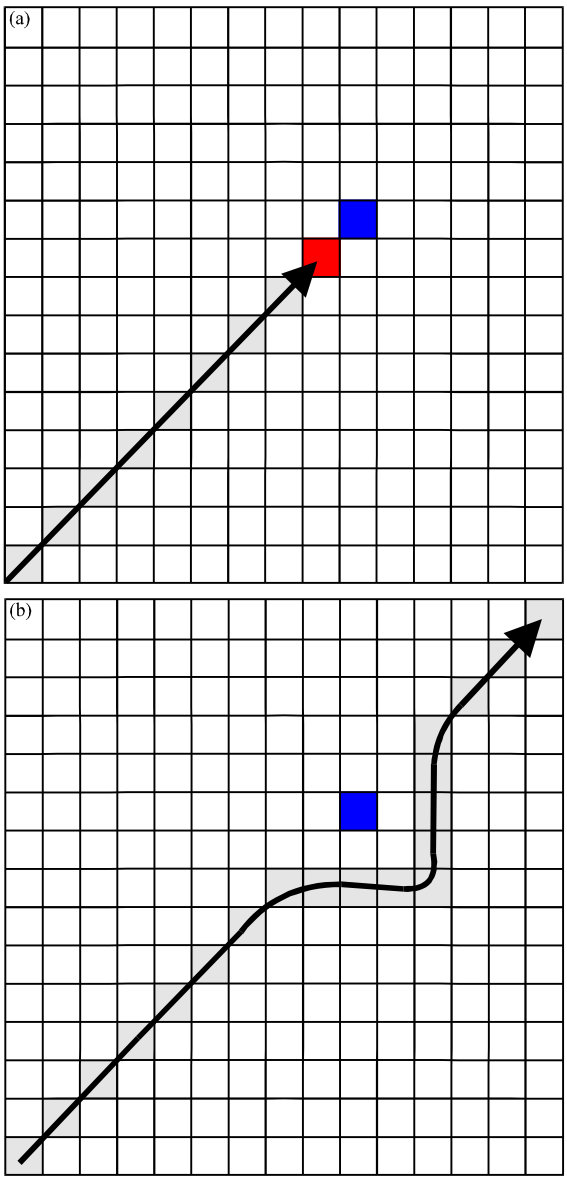

Fig. 7(a-b): Illustration of path correction, (a) Original path to encounter the tower and (b) Corrected path to pass through tower

IMPLEMENTATION AND RESULTS

The implementation of the multi-agent coordination mechanism and message transmission mechanism are based on the system structure and analysis of agents described in the previous sections. The path correction and dynamic enemy generation are based on the multi-agent coordination mechanism, as shown in Fig. 8.

The class diagram of agents is shown in Fig. 9. The tower and enemy are derived from the base object class that contains generic attributes, such as unique ID and Group ID (GID). For the tower and enemy to quickly react to the environment, a heartbeat function is designed. The agent will compute the current status every time the heartbeat function is called and consider the next step or trigger related events. The frequency of calling the heartbeat must be kept within a reasonable range. Otherwise, the function call will significantly impact performance. Therefore, the delta time of the Frame per Second (FPS) is used to restrict the rate because FPS is related to game performance and the graphics shown to the player will be exactly the same as the current condition.

The result of the path correction and dynamic enemy generation method are shown in Fig. 10. Enemies are generated from the top-left corner and head to the bottom-right corner as their destination. There are also several towers of different types that have been placed on the map. In the first round, the counts of generated enemies are 5 and the weight value of each waypoint is zero. Thus, the shortest path will be chosen as the path of the enemies without avoiding the attack range of tower. As a result, all enemies are destroyed before reaching

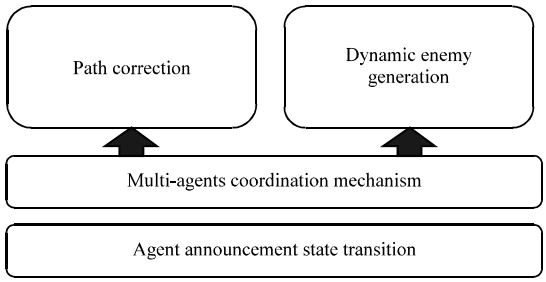

Fig. 8: Function structure diagram

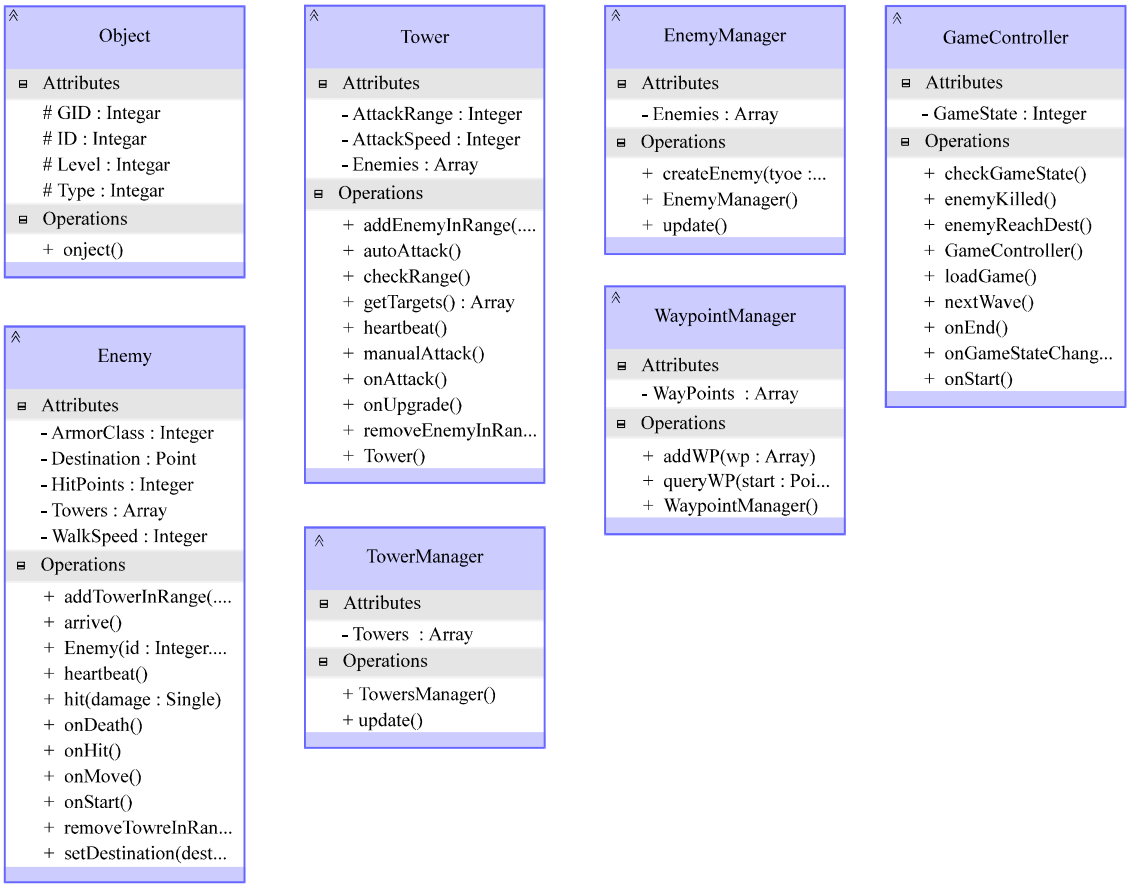

Fig. 9: Class diagram of agents

Fig. 10(a-b): Result of path correction and dynamic enemy generation (a) Enemy path and counts without information from agents and (b) Enemy path and counts with information from agents

their destination. In the second round, enemies change their path to bypass the towers due to the weight values updated from agents that were killed by the towers in the previous round and generation counts increase to 6 because the enemy was too weak to resist attacks. Although, the path of the enemies is rearranged, the 2 player builds a new tower next to the destination in this round and that tower eventually kills all of the enemies

because there is no information about this new tower because its presence was unexpected by the enemy manager.

CONCLUSION

In this article, a tower defense game has been designed and developed based on a multi-agent system and the implemented characteristics of agents to reconstruct several major parts. This paper also presented the model of message transmission and cooperation between agents and described how the task performs more effectively. The results show that the coordination and transmission mechanism of agents changes the game experience of the player and creates a new challenge for this game. Agents will self-learn and react to environment variables to create a better strategy and to compete with the player. In this example, the design is proven to be capable of improving the intelligence of the software and solving our problem. However, this solution also consumed more system resources for computing the logic of the agent. Therefore, performance improvement and the reduction of resource usage are issues to be solved.

REFERENCES

Avery, P.M., J. Togelius, E. Alistar and R.P. van Leeuwen, 2011. Computational intelligence and tower defence games. Proceedings of the Congress on Evolutionary Computation, June 5-8, 2011, New Orleans, LA., pp: 1084-1091.

Balaji, P.G. and D. Srinivasan, 2010. An Introduction to Multi-Agent Systems: Innovations in Multi-Agent Systems and Applications-1. Springer, Berlin, Heidelberg, pp: 1-27.

Bonabeau, E., 2002. Agent-based modeling: Methods and techniques for simulating human systems. Proc. Nat. Acad. Sci. USA., 99: 7280-7287.

Chen, K., H. Wang and H. Lai, 2011. A general knowledge mediation infrastructure for multi-agent systems. Expert Syst. Appl., 38: 495-503.

Chen, M., H. Ishii and C. Wu, 2008. Transportation problems on a fuzzy network. Int. J. Innovat. Comput. Info. Control, 4: 1105-1110.

Cherkassky, B., V. Andrew V. Goldberg and T. Radzik, 1996. Shortest paths algorithms: Theory and experimental evaluation. Math. Prog., 73: 129-174.

Ferber, J., 1999. Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. 1st Edn., Addison-Wesley Longman Inc., New York.

Gallo, G. and S. Pallottino, 1988. Shortest path algorithms. Ann. Operat. Res., 13: 1-79.

Goldberg, A.V. and C. Harrelson, 2004. Computing the shortest path: A* search meets graph theory Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, July 2004, Vancouver, Canada, pp: 1-25.

Goldberg, A.V., H. Kaplan and R. Werneck, 2006. Reach for A*: Efficient point-to-point shortest path algorithms. Proceedings of the Workshop on Algorithm Engineering and Experiments, January 2006, Miami, pp: 1-41.

Gutman, R., 2004. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. Proceedings 6th International Workshop on Algorithm Engineering and Experiments, (IWAEE'04), New York, pp: 100-111.

He, J., H. Lai and H. Wang, 2009. A commonsense knowledge base supported multi-agent architecture. Expert Syst. Appl., 36: 5051-5057.

Horling, B., V. Lesser, R. Vincent and T. Wagner, 2006. The soft real-time agent control architecture. Autonomous Agents Multi-Agent Syst., 12: 35-92.

Ilarri, S., E. Mena and A. Illarramendi, 2008. Using cooperative mobile agents to monitor distributed and dynamic environments. Infor. Sci., 178: 2105-2127.

Ito, N., T. Esaki and N. Ishii, 2002. A cooperative agent model by forming a group. Proceedings of the International Conference on Industrial Technology, Volume 2, December 11-14, 2002, IEEE., pp: 1260-1265.

Kinny, D. and M. Georgeff, 1997. Modelling and Design of Multi-Agent Systems. In: Intelligent Agents III Agent Theories, Architectures and Languages, Muller, J.P., M.J. Wooldridge and N. Jennings (Eds.). Springer Berlin, Heidelberg, ISBN: 9783540625070, pp: 1-20.

Kotz, D. and R.S. Gray, 1999. Mobile agents and the future of the Internet. ACM Operat. Syst. Rev., 33: 7-13.

Lin, F.T. and T.R. Tsai, 2009. A two-stage genetic algorithm for solving the transportation problem with fuzzy demands and fuzzy supplies. Int. J. Innovat. Comput. Info. Control, 5: 4775-4786.

Liu, J. and C. Yao, 2004. Rational competition and cooperation in ubiquitous agent communities. Knowledge Based Syst., 17: 189-200.

Lo, S.K.C., 2012. A collaborative multi-agent message transmission mechanism in intelligent transportation system: A smart freeway example. Info. Sci., 184: 246-265.

Ozen, B., O. Kilic, M. Altinel and A. Dogac, 2004. Highly personalized information delivery to mobile clients. Wireless Netw., 10: 665-683.

Park, S. and V. Sugumaran, 2005. Designing multi-agent systems: A framework and application. Expert Syst. Appl., 28: 259-271.

Robertson, D., 2004. Multi-Agent Coordination as Distributed Logic Programming. Springer, Berlin, Heidelberg, pp: 416-430.

Rybakov, V., 2009. Logic of knowledge and discovery via interacting agents: Decision algorithm for true and satisfiable statements. Inform. Sci., 179: 1608-1614.

Shah, N., R. Iqbal, A. James and K. Iqbal, 2009. Exception representation and management in open multi-agent systems. Inform. Sci., 179: 2555-2561.

Sycara, K. and D. Zeng, 1996. Coordination of multiple intelligent software agents. Int. J. Cooperat. Infor. Syst., 5: 181-211.

Tornero, R., J. Martinez and J. Castello, 2012. Computing real-time dynamic origin/destination matrices from vehicle-to-infrastructure messages using a multi-agent system. Highlights Pract. Applic. Agents Multi-Agent Syst., 156: 147-154.

Tweedale, J., N. Ichalkaranje, C. Sioutis, B. Jarvis, A. Consoli and G. Phillips-Wren, 2007. Innovations in multi-agent systems. J. Network Comput. Appl., 30: 1089-1115.

Wooldridge, M., 1997. Agent-based software engineering. Proc. Software Engin., 144: 26-37.

Wu, D.J., 2001. Software agents for knowledge management: Coordination in multi-agent supply chains and auctions. Expert Syst. Appl., 20: 51-64.

Yau, S.M.T., H.V. Leong and A. Si, 2003. Distributed agent environment: Application and performance. Inform. Sci., 154: 5-21.

Yershov, D.S. and S.M. LaValle, 2011. Simplicial dijkstra and A* algorithms for optimal feedback planning. Proceedings of the International Conference on Intelligent Robots and Systems, September 25-30, 2011, San Francisco, CA., USA., pp: 3862-3867.

Zambonelli, F., N.R. Jennings and M. Wooldridge, 2003. Developing multiagent systems: The Gaia methodology. ACM Trans. Softw. Eng. Methodol., 12: 317-370.

Zhan, F.B., 1997. Three fastest shortest path algorithms on real road networks: Data structures and procedures. J. Geograph. Infor. Decisi. Anal., 1: 69-82.