J. Comput. Educ.
https://doi.org/10.1007/s40692-018-0119-8

Chain of command in autonomous cooperative agents for battles in real-time strategy games

Damon Daylamani-Zad[1] · **Letitia B. Graham**[2] · **Ioannis Th. Paraskevopoulos**[3]

Received: 15 April 2018 / Revised: 1 August 2018 / Accepted: 10 September 2018

Abstract This paper investigates incorporating chain of command in swarm intelligence of honey bees to create groups of ranked co-operative autonomous agents for an RTS game in to create and re-enact battle simulations. The behaviour of the agents are based on the foraging and defensive behaviours of honey bees, adapted to a human environment. The chain of command is implemented using a hierarchical decision model. The groups consist of multiple model-based reflex agents, with individual blackboards for working memory, with a colony level blackboard to mimic the foraging patterns and include commands received from ranking agents. An agent architecture and environment are proposed that allows for creation of autonomous cooperative agents. The behaviour of agents is then evaluated both mathematically and empirically using an adaptation of anytime universal intelligence test and agent believability metric.

Keywords Real-time strategy games · Swarm intelligence · Agent architecture · Multi-agent intelligence · Artificial intelligence · Chain of command

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s40692-018-0119-8) contains supplementary material, which is available to authorized users.

✉ Damon Daylamani-Zad
d.d.zad@greenwich.ac.uk

Letitia B. Graham
letitiagraham@hotmail.co.uk

Ioannis Th. Paraskevopoulos
ioannis.paraskevopoulos@altran.com

[1] Department of Computing and Information Systems, University of Greenwich, London, UK

[2] Bossa Studios, London, UK

[3] Altran Italia, Milan, Italy

Published online: 26 September 2018

Introduction

Most genres of video games require a degree of artificial intelligence (AI) either as support with progress or as opponents. These could be central intelligences, individual agents or groups of agents. Specific genres of games require large numbers of individual AI units to work together for or against the player, such as in real-time strategy (RTS) or Tower Defence games. Taking advantage of the ever increasing computer processing power, it is possible to use more complex algorithms and techniques to create more realistic and challenging AI with less computational expense. Multi-agent approaches and swarm intelligence are inspired on the ability of social animals and crowds to work together as a group without the need for a leader to delegate tasks. Individuals in a swarm are unable to find a solution to a colony's problems alone; however, by interacting with each other and making decisions based on local information, they can find a solution at the colony level (Garnier et al. 2007).

In AI, swarm intelligence aims at creating a decentralised group of autonomous, self-organised individuals that respond to local stimuli. When these individuals are viewed at the swarm level, individual decisions should be contributing to the appearance of a group decision. Several algorithms are inspired by biological swarms; e.g. particle swarm optimisation is based on birds flocking (Kennedy and Eberhart 1995) and ant colony optimisation is based on ant foraging methods (Dorigo and Blum 2005). Swarm intelligence does not tend to have leadership or chain of commands in their implementations, yet battle management is an essential part of an RTS which is not fully implemented in swarms. The concepts of rank and chain of command are integral to realistic implementations as they would allow for the planning, coordination, and monitoring of units through leaders (Løvlid et al. 2017).

The aim of this research is to implement chain of command into an adaptation of the real-life swarm intelligence of honey bees, creating a group of co-ordinated AI agents for an RTS game setup. The goal is to recreate autonomous AI that can be used to re-enact battles in a simulation format, which could also be used in history or military training. This work builds on previous implementation of Cooperative agents (Daylamani-Zad et al. 2017) and addresses the challenge of command structure, allowing for new outcomes such as retreating and surrender.

This paper uses elements of pre-existing algorithms, with the aim of implementing chain of command in a multi-agent system based on behaviour of honey bees. The independent behaviour of agents aims to simulate an over-arching strategy under a hierarchical decision making-approach. "Approaches in games" presents instances of swarm intelligence and chain of command used in games; then, in the following section, the complexity of the problem and the research overview are presented. The behaviour and roles of honey bees are presented in "Bee colonies as a basis for unit development" and existing approaches in multi-agent systems are presented in "Multi-Agent systems". "Mapping the bee behaviour" presents the proposed mapping of bees to soldiers, setting the ground for presenting the agent architecture and the proposed environment. The evaluation

section shows the results of experiments and present an evaluation method based on anytime universal intelligence test. Finally, the paper concludes in "Conclusion".

Approaches in games

Multi-agent approaches including swarm intelligence have been used in games in recent years. These approaches have been used in both serious games used for military training, problem solving in battlefields, as well as used in traditional video games for entertainment.

Reynolds (1987) proposed an approach in simulating the flocking behaviour of birds, creating a distributed behavioural model much like that at work in a natural flock as an alternative to scripting the path of each bird individually for animation. Li and Hu (2009) present a soccer simulation implemented using a multi-agent approach that implements a blackboard model to tackle the communication between the agents. Other researchers (Orkin 2011; Orkin and Roy 2009) have used multi-agent blackboard approach to creat autonomous NPCs (Non-Playable Characters) that behave similar to humans to improve NPC behaviour and increase game engagement. Particle Swarm Optimisation has been used in a tower defence game to optimise cannon locations to cause enemies the most damage (Huo et al. 2009). The scenario contained two teams of players, attack and defence. The attack team needed to navigate along a single path that ensured minimum casualties. The defence team was given seven cannons to place on the map, with the aim of causing the maximum amount of damage to the enemy regardless of the path they took.

An adapted version of ant colony optimisation (ACO) was used to simulate resource gathering in a real-time strategy game environment that was believable to players. A memetic ant colony system was created, using ACO to explore the environment and communicate information about resources. This solution was chosen as it is less computational intensive than using explicit planning and searching. Experiments using this system with different levels of difficulty found that agents in this system were successful even when there were many obstacles and few resources available (Chen et al. 2013).

Stanescu et al. (2014) present a multi-level abstraction framework for RTS multi-agent systems using hierarchical adversarial search framework to more closely model the human way of thinking similar to the chain of command employed by the military. At each level they implement a different abstraction level of decisions which starts from how to win at the top level to individual unit orders. They apply their framework to a StartCraft combat simulator and evaluate it against existing approaches with promising results.

Complexity of agents in RTS games

RTS games comprise of a variety of units and roles, traditionally militaristic, controlled by players to achieve the final goal of the game. This goal tends to be waging war, and winning the war against the opponent(s) (Robertson and Watson

2014; Tavares et al. 2017; Yannakakis and Togelius 2015). The complexity of RTS games is enhanced by the fact that player not only has to control the militaristic units but also needs to gather resources to upgrade and develop its military power. These two aspects work in collaboration; the gathering and economical aspect allows for the development of the military aspect, whilst the military aspect defends and protects the gathering aspect against hostilities and enemies. Considering these two aspects, it is possible to conclude that the main aim of the game is actually survival and growth.

Players need to make three levels of decisions on the units; *Strategic decision-making, Tactical decision-making* and *Micromanagement*. *Strategic decisions* are high-level, long-term planning and decisions which involve the high-level goal of growth and survival. *Tactical decisions* are medium-term plans and are aligned with the strategic decisions but are more detailed, and, for example, can deal with collective actions of a group of units. Finally, *Micromanagement decisions* are short term and relate to controlling individual units (Buro and Churchill 2012; Robertson and Watson 2014).

It is clear that RTS games require multiple levels of abstraction and reasoning within a vast space of actions and states. Humans have a much higher ability to abstract, reason, learn and plan compared to AI (Buro and Churchill 2012; Robertson and Watson 2014). Most well implemented agents are outmatched by experienced human players (Synnaeve and Bessière 2011; Weber et al. 2010).

This paper suggests that using the crowd approach in implementing the AI would potentially bring the intelligence of the AI closer to that of an experienced human player. The approach is inspired by the existing structures within military with chain of command where the level of abstraction in decision-making increases as the rank goes higher. The units make short-term micro-decisions within the parameters of the commands they have received whilst the high-ranking officers would make the strategic decisions. This approach aims for believable AI and believable agents (Yannakakis and Togelius 2015), aiming to create agents that exhibit believable human-like characteristics. Various swarms such as bees, ants, birds and wolves present a potential opportunity to create simple unit agents which make decisions and perform tasks individually within an over-arching strategy that as a whole exhibits an intelligent overall design.

The nature of RTS games as presented requires AI agents which exhibit believable characteristics but it also requires coordination and collaboration. The complex nature of RTS games is one of the greatest challenges in multi-agent systems. Gathering and using resources, waging war and defending friendly units to achieve an over-arching goal are the basis of decision-making, task allocation and action selection in RTS games (Tavares et al. 2014, 2017).

Considering these arguments, it is possible to identify the challenges of implementing believable agents for an RTS game as a combination of choosing a suitable swarm approach which would then need to be mapped to a multi-agent system architecture. Finally, the parameters of an RTS game would need to be mapped to the characteristics of a swarm for implementation. Figure 1 illustrates the workflow of the research as is presented in this paper. The discussions start with establishing the honey bee swarms as the basis for the design of units and their

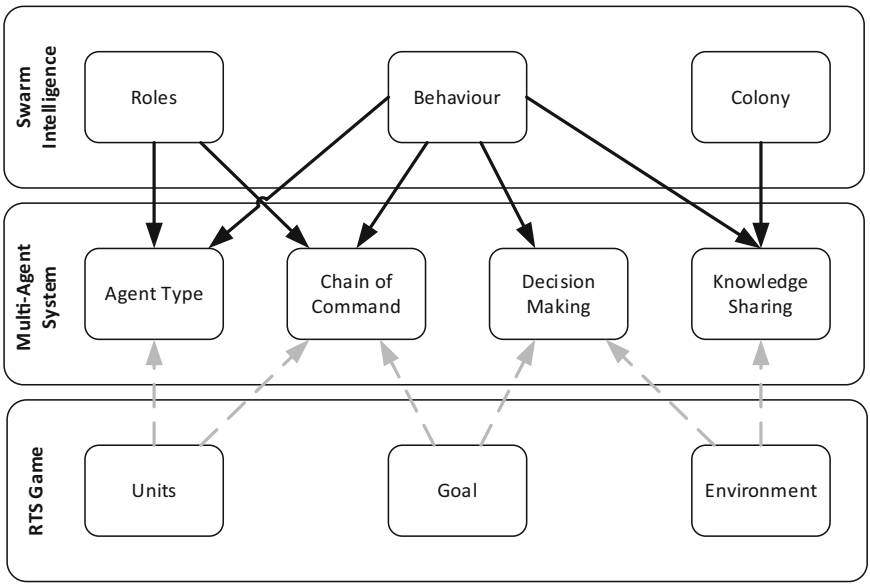

Fig. 1 Research workflow for believable AI agents in an RTS game using swarm intelligence approach

behaviour and roles within a colony. The following section will look at the implementation of a multi-agent system that can incorporate the required elements of an RTS game where units are based on bees. Following that, an implementation is provided where the behaviour of bees is mapped to units and the agent architecture for such system is presented.

The discussions presented identify three aspects in which the research is focused on addressing. These are unit intelligence, command intelligence and agent believability. The unit intelligence is concerned with how intelligently would the units work at a micromanagement decision level. The command intelligence is concerned with the implementation of a chain of command and how well the units follow tactical and strategic decisions in their task selection. Agent believability is focused on the whole system and how the system is perceived by humans.

Bee colonies as a basis for unit development

The behaviour and roles within bee hives have been used as the basis of the agent behaviours and will be mapped to unit behaviour in an RTS in "Mapping the bee behaviour". Group decisions in honey bee colonies are formed by many individual bees' decisions. According to Detrain and Deneubourg (2008), an individual's decision can influence the decision of others, causing the appearance of a group decision. Each bee would make decisions based on its interactions with the environment and other nest mates. The roles and behaviour within a colony can be closely mapped to units within a strategy battle game.

Behaviour of honey bees

Bees exhibit two main behaviours that are useful in a strategy battle games. These are **Foraging** and **Defensive** behaviour.

Foraging

When foraging, honey bees actively recruit others to food sources, providing information about the source through a waggle dance. Waggle dances consist of a series of waggle runs followed by a semi-circular turn, communicating the distance, the angular location based on the sun's azimuth and the odour of food. These dances provide bees with positive feedback that influences others to go to certain locations. The waggle dance is also used by bees to tell nest mates about suitable nest locations when a swarm is looking for a new home (Menzel 1990; Seeley et al. 2006).

While searching for the flowers, bees will take an irregular path and possibly fly hundreds of meters from their hive; however, when they fly back to their nest after locating a food source, they take the path with the shortest distance (von Frisch 1965). Experiments have shown that bees can learn and make decisions based on visual stimuli using their own working memory (Zhang et al. 2005). This working memory is what allows bees to navigate their environment and call on experience to make decisions about the profitability of food.

The foraging behaviour forms the basis for resource gathering and exploration behaviour in a strategy game. The units would need to gather resources and also explore the map to identify new resources or enemy units. The waggle dance is a good map to the reporting behaviour that units would exhibit in RTS games.

Defensive behaviour

The defensive behaviour of the colony is also viewed as a collection of individual responses to stimuli from the environment, such as recruitment pheromones from colony members. Defensive responses of a bee can be broken down into four sequential phases: alerting, activating, attracting and culminating illustrated in Fig. 2 Repeated disturbances of the colony can invoke a fifth phase called absconding, whereby the queen and adult bees leave the nest (Collins et al. 1980).

In each of these phases, there are number of actions a bee can take; however, they can only perform one action at a time and each action requires a certain amount of complex available energy (CAE). By reacting to a stimulus, a bee can step through each of these phases and take an action. There are multiple forms of stimuli that can invoke defensive behaviour, including moving visual stimuli, vibrations of the nest and the alarm pheromones of nest mates (Hunt 2007).

(a) *Alerting*

In the alerting phase, bees have the options of alert, recruit or flee.

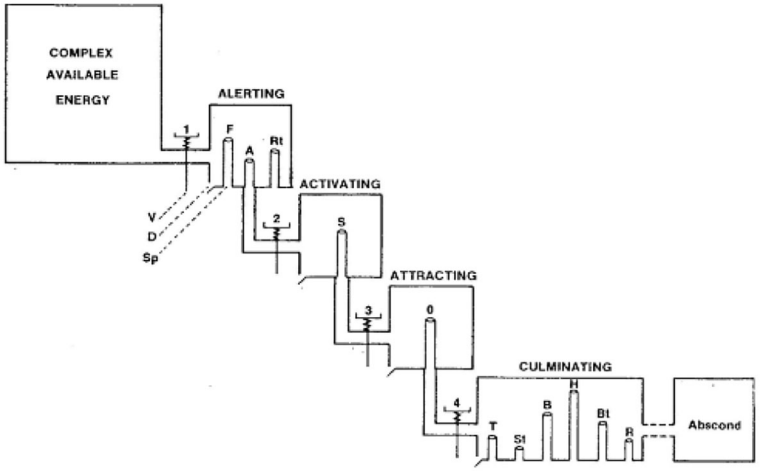

Fig. 2 Model of honey bee defensive behaviour (Collins et al. 1980)

- **Alert** bees take a defensive stance with their wings extended, mouth open and antennae waving. This response is not based on the direction of the stimulus, as it has been found that alert bees that are grouped together face in different directions.
- A **recruiting** bee will open their sting chamber and run into the hive, releasing alarm pheromone to stimulate nest mates into defensive behaviour. A recruiting bee can be recruited into further defensive action by their own pheromone.
- If the stimulus provides directional information, some bees will choose to **retreat** from the area of disturbance.

(b) *Activating*

With more stimulation, bees reach the activating stage. Here, bees will look for the source of the disturbance. Depending on if the hive is opened or unopened, the search will start close to the bee or the hive entrance. If there is no further stimulation after a period, the activated bee may begin searching meters from the hive.

(c) *Attracting*

If an appropriate stimulus is found by an active bee, they will orient towards it. The same disturbance often simultaneously activates and attracts bees. This change in phase is more obvious when a stimulus is presented and then removed to a remote location.

(d) *Culminating*

In this phase, several responses are possible. Bees may sequentially display two or more of the following responses: threaten, run, sting, bite, pull hair or burrow into clothes. If the integrity of the nest has been disrupted, the bee may choose to run.

Roles within a bee colony

During a worker's lifetime, they will undertake different roles as they develop and age, also known as age-related polytheism (Hunt 2007). The main roles they take are: Middle-Aged Bees, and Foragers.

Middle-Aged Bees (MABs) develop around the age of 12–21 days and remain in at this stage of development for a little over a week. MABs can take on numerous roles within the nest. Younger MABs tend to take on tasks such as comb building and colony maintenance, while older MABs take on tasks closer to the hive entrance such as nectar receiving/processing and guarding (Johnson 2010). Around 10–15% of MAB workers will take part in guarding (Hunt 2007). The rest of the MABs tasks are closely related to foragers. Bees can choose which task to do based on feedback from the local environment, e.g. foragers performing the tremble dance can recruit MABs to act as receivers (Detrain and Deneubourg 2008).

Foragers Once a bee has developed into this role, they no longer take part in hive-related tasks that MABs handle. Instead, foragers focus on collecting all resources (Johnson 2010).

There are two more roles which have not been used in this work. The **Cell Cleaner** role refers to newly emerged bees that have not been involved in the nest duties yet. The **Nurse** role is responsible for feeding the young and caring for the Queen. Instead, there are two leadership roles that will be introduced in Sect. 5.26.3).

Multi-Agent systems

Swarm Intelligence can be implemented as a multi-agent system. A multi-agent system consists of an environment which multiple AI agents communicate and act within. An AI agent is described as something that can perceive its environment through sensors and make decisions/act based on information found in that environment (Russell and Norvig 2009). An agent is expected to be able to:

- Operate autonomously
- Perceive their environment
- Persist over a prolonged period
- Adapt to change
- Create and pursue goals

Rationally, agents should aim to select an action that is expected to maximise its performance measure, based on evidence provided by the percept sequence and whatever built-in knowledge the agent has (Russell and Norvig 2009).

Model-based agents

There are different agent types that can be created to replicate the behaviour of the honey bees. The type of agents depends on the environment it interacts with, the tasks it needs to achieve and the process in which it would need to reason. The environment in which the units are interacting with in a battle scenario is partially observable, sequential and dynamic. The agents would need to make a decision regardless of the amount of information they have; therefore, even if there is uncertainty, they would need to make a decision regardless.

Model-based reflex agents are considered the best fit for implementing the agents used within such environment (Lieck and Toussaint 2016). These types of agents keep track of part of the world they cannot see anymore, using information perceived historically. The state of the world the agent is tracking can be updated using information about how the world evolves independently of the agent, and how the agent's actions affect the world (Miller et al. 2016; Scheidt and Pekala 2007). This type of agent can maintain an internal state of the world that is dependent on the percept history. To be able to update this internal state, the agent needs to know how the world works, known as the model of the world.

The model that agents use needs two types of information:

- How the world evolves independent of the agent
- How the agent's own actions affect the world

Using this state and the current percept, the agent can make decisions as to what to do. As the environment is partially observable, the state maintained by the agent is better thought of as a best guess. This means there is likely to be uncertainty in the state; however, decisions still need to be made (Miller et al. 2016; Russell and Norvig 2009).

Blackboard architecture

A blackboard is a global structure that is available to all agents in a system to share information and collaborate to solve a problem. Traditional blackboard systems are made of three components: the blackboard, several knowledge sources (KSs) and a controller component. The blackboard acts as the shared memory for the KSs to read and write from. A KS is a system that can read information from the blackboard, process anything relevant to it, and contribute information towards solutions. KSs are independent of each other, and each can be a different type of system, allowing different approaches to problem solving; however, only one KS can write to the blackboard at a time. The controller component is responsible for choosing which KS can write to the blackboard next. Decisions are made based on

what the KS will contribute and the resources required to create this contribution (Corkill 1991).

More recently, a different approach (Corkill 2003) was suggested for blackboards in a multi-agent system. Corkill proposed a system whereby each agent has their own blackboard and all KSs used in the system, allowing them to focus on nearby data and share with other agents they meet. This approach would create a flexible distributed system, similar to how honey bees use local information and their own experience to make decisions and share information. This has been taken on by other researchers and expanded to using a private and a shared blackboard, where part of the world is shared amongst the agents and other information are kept private to each individual agent (Orkin and Roy 2007). The use of private and public blackboards enables the communication between agents to be precise so that public information is available to all agents through call outs and the nest. The private blackboard would be able to hold information specific to the units which could include their current view of the environment that has not been shared with the nest or commands from ranking units.

Markov decision process for decision-making

AI agents will need to be able to make decisions about the tasks they are performing, the information they share with others, enacting orders, the actions they will take during defensive behaviour and the nodes they will travel to, during resource gathering. Various methods can be used to enable agents to make these decisions. Due to the probabilistic nature of bees' foraging patterns and the model of defensive behaviour, the Markov Decision Process (MDP) is rendered to be the most suitable.

MDP is a process where all possible states are known, each state having related action and reward values, and a probability of transitioning from one state to another (Mausam and Kolobov 2012). MDPs are most suitable to scenarios whereby the agent moves through several known sequential states, where transitions between states happen via a decided action. These points, where decisions are made, are known as decision epochs. Taking an action results in a reward value, which can be positive and negative, with negative values being a cost, rather than a reward. Agents know the rewards for actions before they are taken.

To handle a partially observable environment, Partially Observable MDPs (POMDPs) can be implemented. POMDPs have a similar structure to MDPs; however, they require a sensor model to be able to create a belief state; a group of actual states the agent might currently be in (Russell and Norvig 2009).

Chain of command

In a realistic battle scenario, there is a need for battle management which would require analysis of the state of battle, disposition of units, the health and state of various units, enemy units' state and disposition and the overall tactical planning of the battle. In human terms, this is achieved through the implementation of chain of command (Churchill and Buro 2012; Robertson and Watson 2014).

There have been multiple approaches to implement a command structure. Majority of approaches (Løvlid et al. 2017; Stanescu et al. 2014) implement a hierarchical structure in which agents are ranked and, therefore, could have subordinates that follow their command, whilst, in turn, they would themselves be subordinated to higher-ranked agents which they receive commands from. Such approaches dispense the planning and tactical thinking into multiple levels at unit level where various units would make tactical decisions at different abstraction levels based on their rank within the hierarchy. Notably, Mock (2002) and Rogers and Skabar (2014) have proposed approaches which creates multiple abstraction levels where the high level strategic planning is propagated to units.

Yet, these approaches do not usually include the high-level commanding agents as units in the field. The proposed approach in this paper aims for a platform that enables battle re-enactments and simulations mainly for research and educational purposes and, therefore, needs all the commanding units to also be included in the field of battle.

Mapping the bee behaviour

The prior sections investigated the behaviour of honey bees that will be adapted for use in an RTS AI agent. This section discusses the findings and the proposed approach to implement a hierarchical multi-agent framework within the game to simulate battle scenarios.

As previously mentioned, a bee, during its lifetime, will undertake several different roles. In the game environment, there will be several different roles that AI can undertake and switch between depending on the needs of the group, e.g. Harvesters, Foragers, and Soldiers. These roles are then mapped and simulated into agents as well as leadership roles which will be introduced further on.

To replicate the bees working memory, each agent in the system will have their own private blackboard to store locally perceived information to be used to make individual decisions. To best use this working memory, model-based reflex agents will be implemented to keep track of the environment based on the models of honey bee behaviour discussed earlier. Based on the abstraction level of the agents, the information stored in the memory would be different. The leader units would have a higher abstraction of information with a wider view of the environment which will be populated by the reports from subordinates. The bottom-level units would have a highly granulated view of the environment but in a limited range depending on the information received from ranking units or the nest.

The foraging behaviour is implemented into the game environment, using a shortest path calculation based on Dijkstra's algorithm to wander in the map and locate resources and enemy targets. Once resources are found, the units would move directly towards the base to deposit their load and will continue collecting using this new direct route. On returning to base, they will be able to access a global blackboard to which all agents can read and write. A successful forager can then add the location of the resource found to the blackboard, describing how to get to a resource in a similar way to the honey bee's waggle dance. The units will use this

point as a target direction for gathering resources. Negative feedback will occur when resources are depleted; the path is removed from the blackboard and other resources will need to be found. If foragers encounter enemy targets such as enemy units or the enemy base, they would report this to the closes ranking officer, normally their direct captain. The captains would then decide to engage in battle, report to their leader, general, or ignore the target for the time being but keep track of their movements.

If the nest is disturbed, units undertake defensive actions which are presented as the model for an individual bee. Bees step through the same phases of behaviour, however, can take different actions in each phase dependant on how they perceive the disturbance. To handle the probabilities involved in choosing actions in defensive behaviour, a Markov decision process has been implemented. The process allows agents to select different actions to move from phase to phase, depending on the agent's remaining stamina. MDP could also be adapted to handle the decisions of which node to move to during foraging, dependant on the paths that have recently been travelled along.

Environment and setting

The simulation scenario involves a randomly generated environment that holds two bases (nests). This map hosts two separate colonies known as the Defenders and the Aggressors. There are resources scattered around the environment which the Defenders aim to collect as their base would produce a new agent after each ρ amount of resources collected. Aggressors would be scouring the environment, hunting Defender agents and searching for the Defender's base. This environment is created inspired by Λ environment (Chmait et al. 2016; Insa-Cabrera et al. 2012) which is one of the environment classes that implements the theory behind the Anytime Universal Intelligence Test (Hernández-Orallo and Dowe 2010).

Once the aggressors have spotted the defenders', base, they will stop foraging and gather to attack. At the same time, as the aggressors have also been spotted, the defenders would issue a "Call to Arms" (alert- $>$ recruiting) which would result in recruiting the defenders into a defensive position ready for the imminent attack from the aggressors. The simulation will arrive at its conclusion when both sides line up and charge each other resulting in battle. Once a side is completely wiped out, the simulation ends.

The resources within the map are used for replenishing the health and the energy of defenders. The collected resources will also allow the defender's castle to build new defender units. The aggressor's start the game with existing resources presumed to have been gained from previous raids.

Roles

The roles within the simulation are divided into three categories; the roles specific to defenders, roles specific to aggressors and leadership roles shared by both groups.

(1) *Defender roles*

- Harvesters (Foragers): collect resources only, cannot attack but can alert the castle about attackers. The have Low CAE.
- Militia (MAB): collects resources, but can switch to defensive mode to protect castle when called.

(2) *Aggressor roles*

- Scouts (MAB): do not collect resources, only look for defenders or the castle, can attack when called to arms, can travel further than Attackers and can detect enemies/castle in a wider range.
- Attackers (MAB): do not collect resources, can look for defenders/castle, can attack but cannot travel as far as Scouts unless attacking a castle.

(3) *Leadership roles*

- General: is the first agent for each team that's deployed when the simulation is starting
- Captain: created for every five agents deployed for the team, e.g. 5th, 10th, 15th.

The General and Captain roles have all the functionalities of the harvesters/militia/attackers. In addition, they have additional functionality; they are able to call for their whole team to *retreat* or *surrender*. These two roles would take the characteristics of normal roles within the team but they are also officers and, therefore, would be able to exhibit actions and make decisions that other units cannot.

Agent architecture

The proposed architecture for the agents in presented in Fig. 3. The agent consists of blackboard and processing components. Information is received by the sensor form the environment, this information is then stored in the blackboard. The Action Selector unit would than select a suitable action based on the information on the blackboard. This final decision is then passed on to the actuator which would apply this to the environment, hence, exhibiting a behaviour.

The blackboard is divided into two sections; a private blackboard and a shared blackboard. The private blackboard is private and independent for each agent. The information on the private blackboard are only accessible to an instance of the agent and relate to this agent. The private blackboard would include information such as the rank of a unit, the status of troops (its subordinate agents, in case the unit is a captain or a general) and the direct commands it receives. The shared blackboard is information that is public amongst all agents. This simulates the knowledge held at

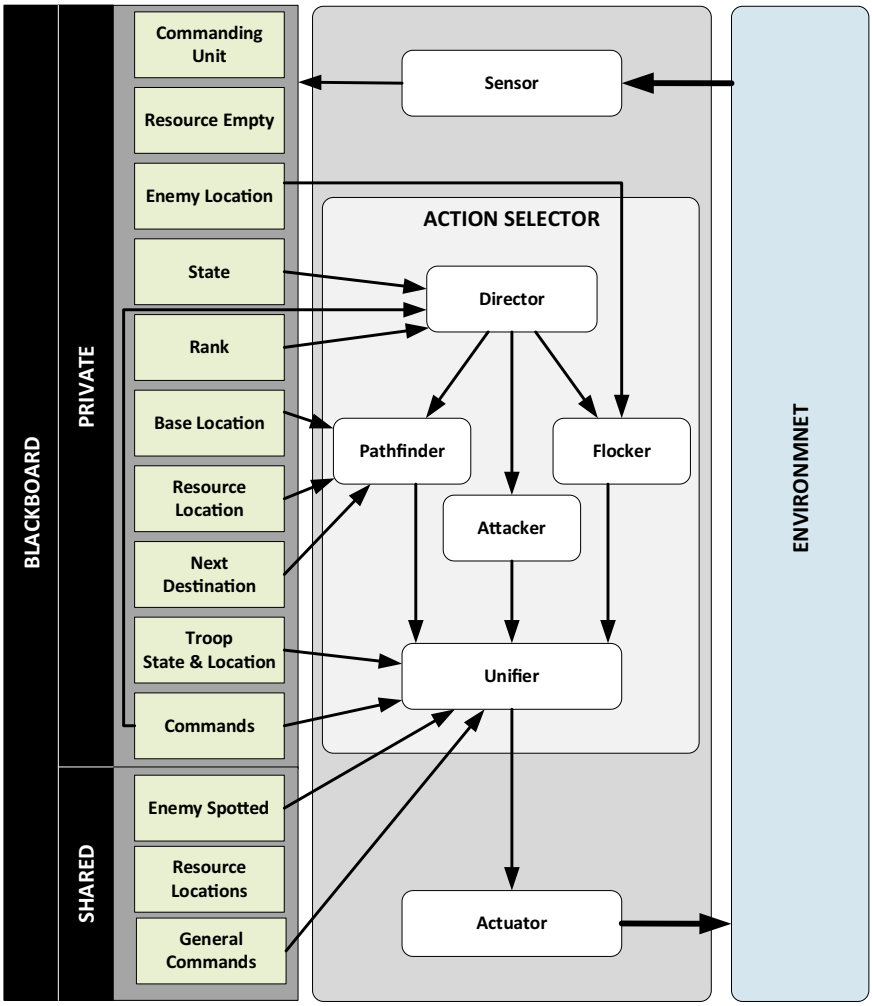

Fig. 3 Agent architecture

the nest that is publicly known to all agents. The commands from General which applies to all units are available in the shared blackboard.

The action consists of a director module that reads the current state of the agents, and then based on the current state of the world and the current command issues to the unit, decides which of the underlying experts are needed to make the next decision. At any given situation, the agent might need to take multiple decision through multiple experts. The agent might need to move and flock at the same time as flocking action would involve moving and, therefore, pathfinding. The three expert units; pathfinder, attacker and flocker will use the information from the agent state and the world state and make their own relative decisions. Finally, the decisions made by the three experts are passed to the unifier module which would

consider the current command issues to the agent as well as the status of its subordinates (if any) and then unify these into a decisions which would formed into actionable directives that are passed to the actuator.

The director and the three expert modules base their decision-making on a hierarchical state machine of the agents that represents the states which the agents would go through based on the information received from the game, commands issues to the agent and the decisions that the agents have taken. Figure 4 represents the high-level agent state machine. Each agent after it is created would go through a setting up state where its role and initial action are decided. Once the agent is ready, it will go to the foraging state where as the aggressor or defender, it will start to look for defenders or resources, respectively. From foraging, if they meet enemies, they will go to attacking or defending states and finally when an agent's health reaches zero, it will die. During foraging or conflict states, if the agent is a leader, i.e. a General or a Captain, then the agent will continuously go into assessing state which would assess the battle and decide on retreating or continuing the previous state for its subordinates (Troops). Once Retreating call has been issued, all troops/units involved in conflict would return to their respective bases and not return. If the highest-ranking leader has reached the base and is still in retreat, then a Surrender call is issued. Surrender stops the agents of the team from acting any further and changes their colour to white; hence, the simulation ends at that point.

The foraging state, illustrated in Fig. 5a, encompasses the state machine for movements and path finding. This state machine would allow foraging unless an enemy or resource has been spotted in which case the agent would move to the conflict state or focus of collecting resources. As presented in Fig. 5b, conflict state

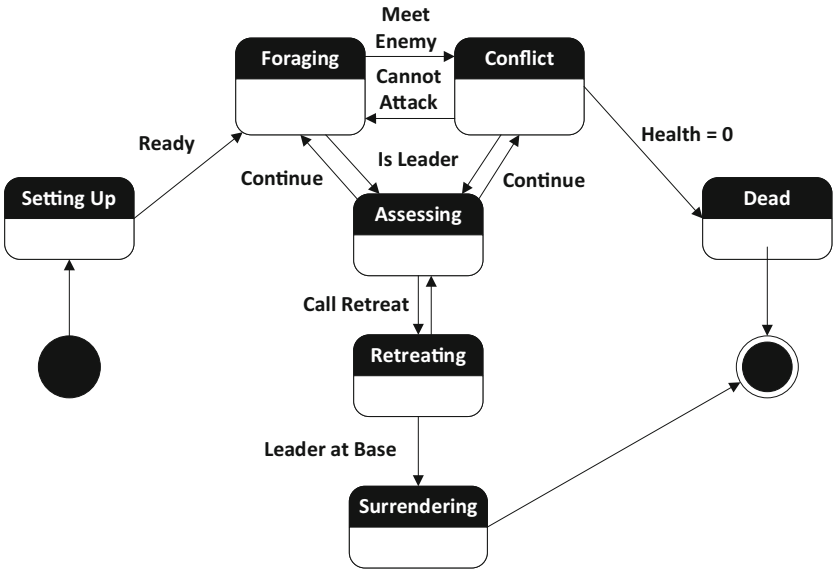

Fig. 4 High-level agent state machine

controls the decision for the various agent roles during the disturbances. The aggressor agents will call to arms all their fellow attackers and scouts, flock to position and charge the defenders. The defenders on the other side will get into formation, and charge the aggressors to defend their castle. There is a chance that the harvesters might flee to base instead of joining the defence. The aggressors have a special state, Hunt, as they would hunt for enemies if they are not attacking the castle. The flock to position state deals with the flocking behaviour of the agents. The state machine, presented in Fig. 5c, allows the agents to make decision on their positioning, when responding to call to arms, or when readying to get in line for charge. The flocking behaviour is at the core of crowd and swarm intelligence. The flocking patterns are a great demonstration of independent agents behaving separately and yet creating a crowd pattern. The assessing state, Fig. 5d, illustrates the process in which the retreating or continuing decision is made. The Decide

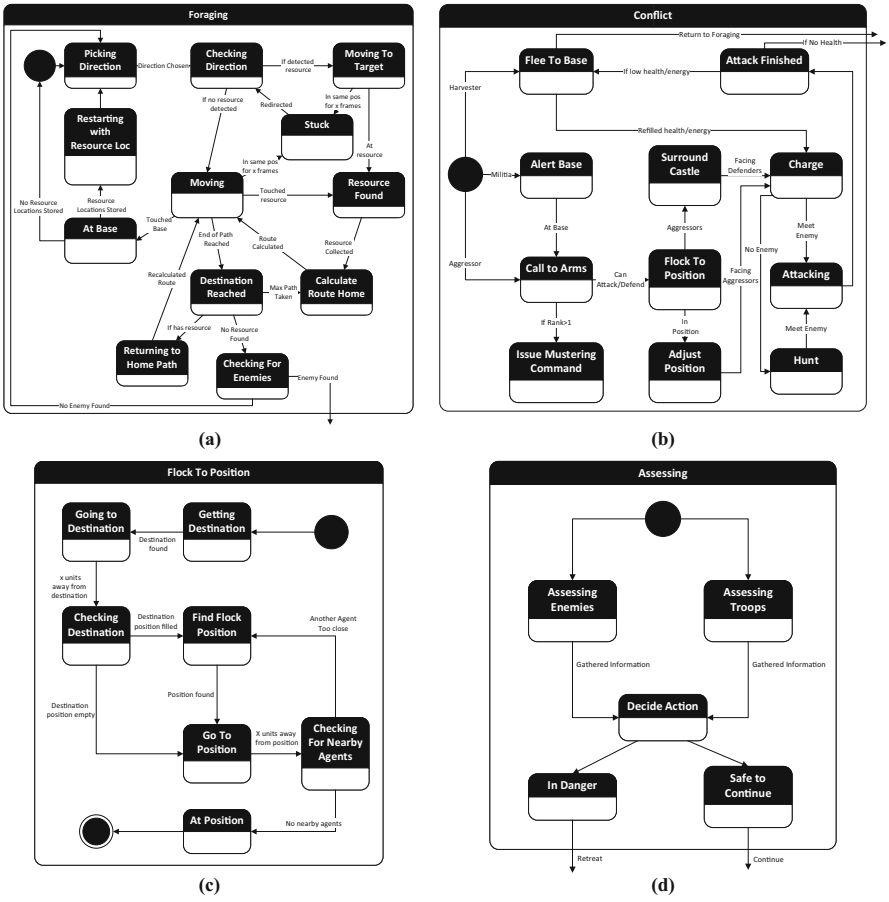

Fig. 5 States of the agent: **a** foraging state, **b** conflict state, **c** flock to position state (Daylamani-Zad et al. 2017), **d** assessing state

Action state works differently based on the leader unit that is making the decision. Captains would call retreat if one the following conditions are met;

C1. If number of living troops is less than 45% of living enemy troops involved in conflict.
C2. If number of injured troops is less than 50% of injured enemy troops involved in conflict.
C3. If the General is dead.

The troops are the units subordinated to the captain at the time of assessment who are able to attack/defend; therefore, harvesters do not count as troops even though they might be subordinated to a captain. Each unit will be under the command of its closest captain. If during foraging, a unit meets a captain that is closer to it than its current captain (based on the location it has stored), then the unit will switch captains. Also, a unit is considered injured if its health is below 45% of its total health. These units are in danger of dying and therefore if more than half the troops of a captain in conflict are injured, then it is worth retreating and regaining health at the base.

The General would call retreat if one of the conditions of the captains is met or if one of the following conditions is met;

C4. If all the captains are dead.
C5. If all the harvesters are dead.

Table 1 illustrates how the bee behaviour model presented earlier has been mapped to human behaviour during disturbances to the nest. The table shows how recruiting, searching, alerting, attract and culminating will be mapped to simulated human behaviour in the AI units within the game as well presenting the mapping of attack patterns and their weight in the units for the MDP to decide on actions.

If the nest is disturbed, units undertake defensive actions which are presented as the model for an individual bee. Bees step through the same phases of behaviour, however, can take different actions depending on the CAE levels and their role. The agents would use the table above to decide also on the rewards of the actions they take compared to the cost.

Evaluation

This section presents the evaluation of the proposed architecture. The evaluation follows the three aspects identified in "Complexity of agents in RTS games" section; unit intelligence, command intelligence and agent believability. This section presents the methods and the results of the evaluation of each aspect of the proposed architecture.

Table 1 Mapping bee defensive behaviour to unit behaviour (Daylamani-Zad et al. 2017)

Behaviour			Cost (CAE)	Damage
Bee	Human			
	Aggressor	*Defender*		
Recruit	Initiate call to arms	Flee	–	–
	Call to arms at base	Call to arms at base	–	–
Searching	Get in formation	Get into defensive position	–	–
Attract	Get in line for charge	Get in line for charge	–	–
Culminating	Battle cry	Taunt	1	+1 to the next attack
	Sword attack	Sword attack	1	1
	Block with shield	Block with shield	2	No damage
	Shield bash	Shield bash	5	2
	String sword attack	String sword attack	8	4
	Dodge	Dodge	3	No damage

Unit intelligence

The evaluation method used for Unit Intelligence is an expansion of the previous evaluation used by Daylamani-Zad et al. (2017) which was inspired by Chmait et al. (2016b) and their approach to anytime universal intelligence test (Hernández-Orallo and Dowe 2010). For this purpose, the environment is created based on a Λ environment (Chmait et al. 2016a; Insa-Cabrera et al. 2012). The idea is to evaluate an agent that can perform a set of finite tasks based on the environment it is placed in. The intelligence of the system is assessed as a function of successful transition between states and the success of achieving the objective of each state.

Method summary

This section provides a brief description of the evaluation method as presented by Daylamani-Zad et al. (2017). In this approach for evaluating the intelligence of a multi-agent system, each agent has its own specific role and each role would have its own specific set of tasks and states that it is supposed to achieve. Each agent π_i which is a member of $\Pi = \{\pi_1, \pi_{2\ldots}\ \pi_n\}$ has a role ω_j from $\Omega = \{\omega_1, \omega_2 \ldots\ \omega_m\}$. Each role would have a set of states available to them which is defined as $S_j = \{s: s \in States \wedge \omega_j \text{ can be in } s\}$, where *States* is the set including all the states available. An available state s_k would have a good outcome $\oplus_k$ and a bad outcome $\otimes_k$. The good outcome is achieving its objective whilst the worst outcome is the complete opposite. This is a theoretical definition that has been put into practice based on the tasks at hand.

A reward function has been defined for the agents which would represent how an agent is performing in a state based on the best and worst outcomes of that state. The reward of agent π_i in state s_k is represented by $\gamma_{i,\,k}$ where $-1.0 \leq \gamma \leq +1.0$. The value of $\gamma_{i,\,k}$ is calculated using Eq (1) as a function of the outcome of the state

objective, where $f(a, b)$ denotes the success of a achieving b. this can take many forms.

$$\gamma_{i,k} = 1/f(\pi_i, \oplus_k) - 1/f(\pi_\mathrm{i}, \otimes_\mathrm{k}) \tag{1}$$

The intelligence of an agent π_i would be defined as the amalgamation of its rewards in all its states during an iteration of the simulation, denoted as I_i presented in Eq. (2). Hence, the intelligence of the agents in a role, $\omega_\mathrm{j}I$, can then be calculated using Eq. (3) as the amalgamation of all the agents that are acting in the role ω_j.

$$I_\mathrm{i} = \sum_{\mathrm{k}=1\ldots\mathrm{p}} \left(\gamma_{\mathrm{i,k}}\right) / \ \mathrm{p} \tag{2}$$

$$\omega_\mathrm{j}I = \sum_{\mathrm{i}=1\ldots\mathrm{q}} I_\mathrm{i} / \ \mathrm{q} \tag{3}$$

The number p in Eq. (2) is the number of states available for agent π_i and it is important to note that if an agent arrives at, p would not necessarily equal to $|S_j|$. It would depend on the states that the agent has arrived at during a simulation which means some states might be skipped whilst others might have been repeated many times. The number q represents the number of π_i agents that are acting in role ω_j. Finally, the collective intelligence of a set of agents, Π, is defined as $\psi(\Pi)$ and calculated using Eq. (4) as the amalgamation of the intelligence of all the roles. As Ω is the set of all the roles and each agent within Π is mapped to a role, by association, amalgamating the intelligence of each role, would allow for an amalgamation of the intelligence of all agents.

$$\psi(\Pi) \ = \sum_{\mathrm{j}=1\ldots\mathrm{m}} \omega_\mathrm{j}I / \ \mathrm{m}, \mathrm{m} \ = |\Omega| \tag{4}$$

Results

For the purpose of this research, the architecture was implemented using C# and Unity and the simulation was executed for 200 iterations. Each iteration concluded with one side, either aggressors or defenders, defeated and wiped out. The iterations took between 6 and 11 minutes and both sides had an equal number of agents spawned at the start. Figure 6 illustrates the agents in three different states. Table 2 presents the amalgamated rewards for each state for the first ten test iterations, and Table 3 presents the mean rewards and their median and standard deviation for all 200 iterations. There were many instances of the outcome 1 which means that the state has always reached its best outcome and, therefore, illustrates a high intelligence. There are also instances of very low intelligence such as in Surrender where the value has been 0. This can be explained with the fact that a number of iterations did not reach a surrender state or has an unsuccessful surrendering when the simulation ended before a successful surrender. The attack state is also notable for a low score which can be explained due to the nature of attacking

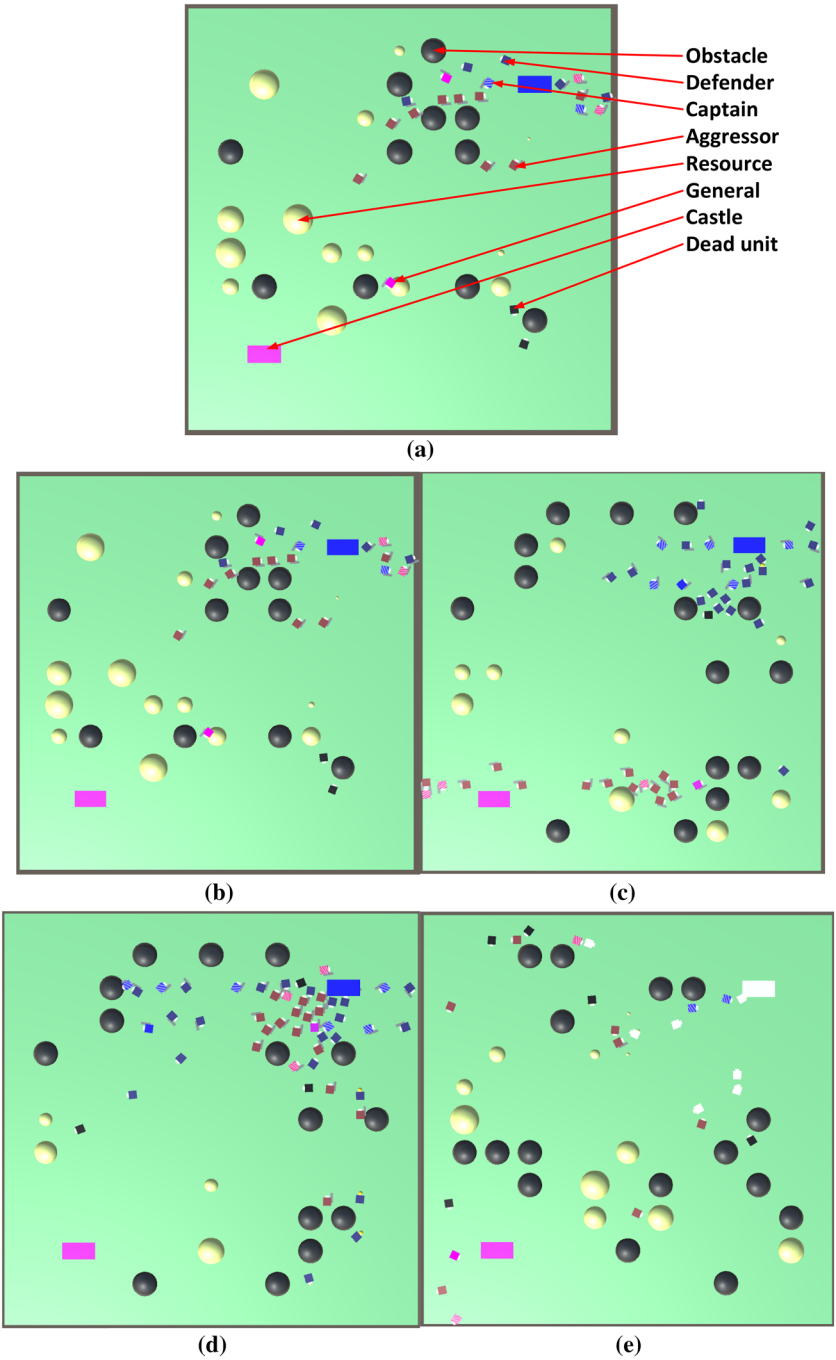

Fig. 6 Simulation of the game. **a** Guide figure, **b** charging the defenders' base, **c** flock to position at base. **d** charge. **e** Surrender

Table 2 Amalgamated reward for each state

State	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
Adjust position	0.8	1	0.81	0.91	1	0.88	0.86	0.92	0.9	0.98
Alert captain	1	1	1	1	1	1	1	1	1	1
At home	0.99	1	1	0.99	1	1	0.99	0.99	1	1
At position	1	0.89	1	0.87	0.87	0.9	1	0.89	1	0.97
Attack	0.7	0.56	0.24	0.94	0.87	0.44	0.37	0.28	0.83	0.57
Attack finished	0.74	0.95	0.94	0.94	0.86	0.93	0.86	0.88	0.78	0.76
Call to arms	1	1	1	0.91	0.91	1	1	1	1	0.91
Charge	0.93	1	1	0.9	0.72	0.98	0.96	1	1	1
Checking destination position	1	0.97	1	1	0.92	1	1	0.92	1	1
Checking direction	0.97	0.99	0.99	0.96	0.86	0.97	0.99	0.99	0.99	0.99
Checking for enemies	0.79	1	1	0.9	0.77	0.94	0.79	0.9	1	1
Find flock position	0.99	0.96	1	0.83	0.88	0.98	1	0.91	1	1
Flee to base	0.05	0.83	0.7	0.94	0.07	0.01	0.2	0.01	1	0.02
Found resource	0.86	1	1	1	0.95	1	1	1	1	1
Getting destination	0.96	1	1	0.87	0.94	1	1	0.95	0.73	1
Get to position	1	1	0.87	0.87	1	1	1	1	0.87	1
Going home	0.88	1	1	0.97	0.83	1	1	1	1	1
Going to destination position	0.96	1	1	0.7	0.74	1	1	0.95	0.83	0
Going to flock position	0.99	0.95	1	0.83	0	0.98	1	0.1	1	1
Hunt	0.92	0.95	0.82	0.84	0.98	0.92	0.87	0.67	0.77	0.87
Moving	0.96	0.99	0.99	0.96	0.58	0.97	0.99	0.99	0.99	0.99
Moving to target	1	1	1	1	0.84	1	1	1	1	1
Pick direction with target	1	1	1	1	0.94	1	1	1	1	1
Picking direction	0.96	0.99	0.99	0.95	0.86	0.97	0.99	0.99	0.99	0.99

Table 2 continued

State	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Iteration 6	Iteration 7	Iteration 8	Iteration 9	Iteration 10
Restarting with resource loc.	1	1	1	1	0.95	1	1	1	1	1
Retreating	0.84	0.92	0.92	0.83	0.91	0.9	0.74	0.85	0.74	0.94
Returning to path home	0.82	1	1	0.97	0.87	1	1	0.96	1	0.98
Starting	0.98	0.87	0.87	0.98	0.98	0.79	0.84	0.9	0.79	0.87
Stuck	1	1	1	0.99	0.92	0.98	1	1	1	1
Surround castle	1	1	1	0.86	0.86	1	1	1	0.99	1
Surrendering	1	1	0	0	0	0	0	0	1	0.69

Table 3 Mean, median and standard deviation reward for each state in 200 simulation iterations

State	Mean	Median	Standard deviation
Adjust position	0.905	0.9	0.065
Alert captain	1	1	0
At home	0.994	1	0.006
At position	0.937	0.92	0.053
Attack	0.597	0.57	0.232
Attack finished	0.866	0.88	0.072
Call to arms	0.968	1	0.042
Charge	0.949	0.98	0.079
Checking destination position	0.978	1	0.031
Checking direction	0.970	0.99	0.036
Checking for enemies	0.909	0.91	0.086
Find flock position	0.949	0.98	0.058
Flee to base	0.429	0.20	0.413
Found resource	0.979	1	0.041
Getting destination	0.947	0.97	0.078
Get to position	0.954	1	0.060
Going home	0.957	1	0.065
Going to destination position	0.820	0.95	0.279
Going to flock position	0.794	0.98	0.355
Hunt	0.863	0.87	0.083
Moving	0.941	0.99	0.115
Moving to target	0.980	1	0.046
Pick direction with target	0.990	1	0.021
Picking direction	0.957	0.99	0.050
Restarting with resource loc.	0.994	1	0.014
Retreating	0.859	0.86	0.065
Returning to path home	0.959	0.98	0.057
Starting	0.883	0.87	0.066
Stuck	0.985	1	0.025
Surround castle	0.969	1	0.053
Surrendering	0.380	0	0.440

behaviour in battle which involves misses, deflections and dodges by opponents as well as unsuccessful starts that might have been stopped due to receiving damage mid-action. Whilst alerting captain seems the most successful state in all simulations, surrendering and fleeing to base have been generally not as rewarded as other states. As mentioned, these can be explained by the fact that these are retreating scenarios when the opponent aims to wipe out to win, which can lead to incomplete execution of these states.

Tables 4 and 5, respectively, represent the intelligence calculated for each role and then the collective intelligence of each team. The intelligence values averaged above 0.78 with maximum deviation of 0.1, the observed intelligence values were generally above 0.7, and this is considered a very high score as the range of possible intelligence would have been between -1 and $+1$. The intelligence scores are close to the top end meaning that the majority of times the agents have been consistently pursuing the good outcome for each state. Whilst, on average, it seems to be the aggressors are performing slightly better that the defenders, it is worth noting that there were many iterations where defenders have performed better than the aggressors. It is also noteworthy that the defenders had a lower deviation and their performance had less fluctuations. This could be explained by the difference in the strategic goal of the defenders which is more focused compared to the aggressors.

Command intelligence

The evaluation of command intelligence aims to evaluate the implementation of the chain of command in the architecture. This evaluation is focused on tactical and strategic decisions made by the commanders and how well these have been followed by the subordinates. This evaluation has two sides; leaders making their respective decisions and units following commands in their actions.

Method summary

To evaluate the command intelligence of the architecture at the leader, the evaluation looks at assessing the following conditions;

- Has the leader made the correct decision based on the situation?

Table 4 Intelligence for each role; mean, median and standard deviation

Role	Mean	Median	Standard deviation
General	0.852	0.855	0.112
Captain	0.826	0.820	0.072
Attacker	0.885	0.870	0.087
Scout	0.894	0.905	0.082
Militia	0.786	0.785	0.083
Harvester	0.918	0.920	0.060

Table 5 Collective intelligence for each team; mean, median and standard deviation

Team	Mean	Median	Standard deviation
Aggressors	0.866	0.870	0.066
Defenders	0.845	0.845	0.026

- How long did it take the leader to issue a command since the occurrence of a command triggering event?

A separate code, named assessment unit (AU), was attached to the simulations which had an encompassing view of all the agents and the environment. The AU would have access to all agents and, therefore, was able to calculate the decisions the leaders should be making, following the exact logic as leaders would. As soon as there was an event that would require a command to be triggered, the AU would start a timer that would stop at the moment the leader issues a command or would stop after 10 s, deciding the leader has failed to issue any commands for the event. The AU has full view as opposed to the partial view of the leaders which also relies on communication between units and, therefore, can make perfect decisions.

To evaluate how units have followed a command (command conformance), the unit decisions are compared with the current command at the time. If the initiated state was in line with the command issued, then it is considered a success; otherwise, it is considered a failure in the command intelligence. There are, however, two levels of commands within the simulations, commands issued by Generals or Captains. The two commands are assessed separately. For example, if a unit's state is in accordance with one but contradictory to the other, then it would be considered positive for one and negative for the other.

Results

The simulations in "Command intelligence" were also analysed in regards to command intelligence. The results of the leader command intelligence are summarised in Table 6. The results show that the delay between the events and commands have been acceptable as they average under 40 ms. It is important to note that the no-command scenarios, where no command was issued after 10 s, are not considered in these averages as they are considered instances of no decision. In regards to the command accuracy, there were no incorrect commands issued. The reason for not reaching 100% accuracy was actually the presence of no-command scenarios where a command was necessary but due to lack of observation or failure in communication, no command was issued. An example scenario was when a unit encountered a group of enemy units but died before an assessment could be made by the leader. The aggressors seem to have slightly quicker and more accurate general

Table 6 Command intelligence for leaders in each team: mean decision accuracy and average decision time

Team	Leader	Command accuracy (%)	Command time delay (milliseconds)
Aggressors	General	89.37	33.52
	Captain	85.76	38.25
Defenders	General	87.33	34.26
	Captain	85.66	36.66

in comparison to defenders but the difference is small and negligible. There are many instances where in an individual simulation the defender general outperformed the aggressor counterpart.

The command conformance data are summarised in Table 7. The mean conformance to commands is above 72% which indicates that nearly three quarters of all commands are fully followed by subordinates. However, this also shows that there is a one in four chance of a command not being implemented by the troops. This is partly due to various circumstances that units might be experiencing. There is the obvious scenario where there are two contradicting commands on the blackboard at the same time. In such a scenario, the agent would have not conformed to one of the two. There are other scenarios where an agent might have started a process that stops them from conforming to a command as the state machine and actuator are unable to find a connecting state that would allow the agent to follow the command received. For example, an agent which is Flocking to Position, must finish positioning and then charge before being able to reassess its situation and follow a retreat command.

The table above also demonstrates that the commands issued by generals have a much higher conformance rate than the commands from captains. This observation is in accordance with the design of the architecture. The architecture has tried to enforce that the general's commands should outrank a captains and, therefore, captain's commands should be in accordance with the general's. The higher conformance mean and median indicate that this has successfully been implemented in the architecture.

Agent believability

This section focuses on the believability of the agents and how well would then be accepted by humans. Whilst the research aims to create a platform for simulation, the acceptance of the simulation by human users would be a key aspect of evaluating the proposed architecture.

Method summary

To evaluate the believability of the agents, an empirical evaluation of the architecture has been implemented based around the believability metrics defined by Gomes et al. (2013). The believability metrics defines eight dimensions to the believability of an agent-based system which an audience can identify. These

Table 7 Command conformance for leaders in each team: mean, median and standard deviation

Team	Leader	Mean (%)	Median (%)	Standard deviation
Aggressors	General	79.73	80	12.85
	Captain	73.50	70	13.07
Defenders	General	80.06	80	13.82
	Captain	72.56	70	17.23

dimensions are; awareness, behaviour understandability, personality, visual impact, predictability, behaviour coherence, change with experience, and social. A questionnaire was created based on the believability metric proposed by Gomes et al. (2013) which would ask participants to answer each question on a Likert scale of five (from 1 = strongly disagree to 5 = strongly agree).

A group of 30 participants (18 male − 12 female) were recruited from undergraduate students of the department of information systems and computing. The participants were aged 18–28 and they all considered themselves gamers and had experience of playing RTS games. Each participant was presented with ten simulation recordings. The recordings were sped up to play six times faster and lasted between 30 s and 2 min and 38 s. Simulations included an equal five wins for both defenders and aggressors. A simulation was included for both sides where one team would surrender. The simulations were not played in the same order for all participants. The participants were not told if the simulation was being played by human players or AI and were free to make their own assumptions. Each participant would watch each simulation recording and at the end would fill in the believability metric questionnaire for both aggressors and defenders. The participants were encouraged to discuss their responses as they fill the questionnaire and these were noted by the researchers for quantitative analysis.

Results

The results of the questionnaire are summarised in Table 8. The participants found the two sides very aware of with a high majority strongly agreeing that the two sides perceived the world around them well. They also agreed that the behaviour of the two sides is understandable under the RTS setting and that both sides behaved cohesively. More cohesion was perceived to be exhibited on the defenders side than aggressor but with a small margin. These results were also supported by the participant comments. Most mentioned that they found the awareness and the response rate very interesting. Five participants mentioned they were very

Table 8 Believability metric questionnaire results

Question	Aggressor			Defender		
	Mean	Median	Standard deviation	Mean	Median	Standard deviation
Awareness	4.43	5	0.935	4.26	4.5	1.048
Behaviour understandability	4.40	4.5	0.813	4.33	4.5	0.884
Personality	4.23	4	0.568	4.40	4	0.563
Visual impact	3.00	3	1.051	3.10	3	1.155
Predictability	4.10	4	0.922	4.73	5	0.784
Behaviour coherence	4.40	4	0.563	4.46	4.5	0.571
Change with experience	3.96	4	1.159	4.03	4	0.808
Social	4.16	4	0.874	3.96	4	0.999

impressed with the retreats as they believed that was the right decision at that time of the game.

The visual impact received mixed reactions. Whilst some participants were very impressed with the agent behaviour and agreed that it draw their attention, others did not like the simple visual design of the simulations and mentioned that the visuals were not up to the standard they were expecting. Yet, other participants which understood the purpose of the question better mentioned examples from the flocking for defence and the charge behaviour as visually impactful for them.

Change with experience included some interesting observations. A majority of participants believed that they are watching a sequence of game-plays and, therefore, assumed that the players were learning from previous game-plays and changing tactics. Another cluster of participants believed that the behaviour of the players did not change even though they were gaining more experience. Both clusters assumed that the players are learning and experiencing multiple game sessions and only disagreed on if there was any improvement/change as a result.

Considering the overall results of the believability metric, the participants received the architecture well and connected with the simulations. The understanding and awareness were specially highlighted by the participants. Considering the decentralised and swarm approach of the architecture, it is possible to conclude that the proposed architecture has achieved believability. Whilst there are areas such as social and visual impact that could be improved, overall, the evaluation against the believability metric has been promising.

Conclusion

This research investigated the idea of incorporating chain of command with swarm intelligence of honey bees when foraging and defending their nests to create a group of co-operative agents with leadership and tactical decision-making. The idea was to create a decentralised group of autonomous agents that could work together to achieve goals that would be found within an RTS game. The goal was to simulate battle behaviour without a central control yet include a hierarchical leadership that would make strategic decision while actually acting as agents within the battle field (Karpov et al. 2012, 2015). Multiple simulations were run and using the recorded data, patterns of behaviour were identified and analysed to check whether the performed behaviour was expected for that role and main state. The analysed data were then used to calculate the intelligence score of each individual agent using an approach based on the Λ environment and the Anytime Universal Intelligence Test. The agents scored consistently above 0.7 in a scale between -1 and $+1$, demonstrating a high level of collective intelligence. The believability of the agents was evaluated through the agent believability metric. The chain of command implemented has produced better results than the previous leaderless implementations and resulted in a more coherent battle scenarios where retreat and surrender were also viable option. This is important in re-enactments and simulations as most

existing computer–based approaches are focused on winning whilst a suitable retreat or a timely surrender are viable decisions in a real-world battle scenario.

A limitation of this research is that so far the system has not been tested against human players or existing RTS AI, as literature suggests this is an interesting challenge within the domain. The aim of the research is not necessarily creating AI that can compete with humans but rather one that can simulate a scenario to a believable level given the parameters. Still, it is an avenue that the researchers plan to pursue in the future. Other future work for this project revolves around scaling up the simulation in terms of environment, roles available in the teams and the action taken during battles. To introduce new behaviours successfully, the environment will need to be larger to handle more agents and more detail. This would allow the system to be expanded and applied to other game genres such as shooter games, racing games and especially serious games and simulations such as rehabilitation games and narrative based trainings. Also, introducing a new role in the defender team called the Scavenger which would act in a similar way to harvesters, yet would take part in battles. They would be able to collect resources from the remainder of dead agents, providing another resource for creating defender agents. Currently, agents can fight on a 1-vs-1 basis, when an enemy is nearby or in a larger battle after spotting the defender base. In a larger environment, it would be possible to call groups of nearby agents together for group battles to occur. If deciding to call other agents for a group battle, agents under the same captain and within a certain radius would create formations and attack each other, hence brining the simulation much closer to true strategic behaviour and battle simulations. The inclusion of nurse roles would also create a more realistic solution which can incorporate the injured units in a more realistic manner.

References

Buro, M., & Churchill, D. (2012). Real-time strategy game competitions. *AI Magazine, 33*(3), 106–108. https://doi.org/10.1609/aimag.v33i3.2419.

Chen, X., Ong, Y. S., Feng, L., Lim, M. H., Chen, C., & Ho, C. S. (2013). Towards believable resource gathering behaviours in real-time strategy games with a memetic ant colony system. *Procedia Computer Science, 24,* 143–151. https://doi.org/10.1016/j.procs.2013.10.037.

Chmait, N., Dowe, D. L., Li, Y.-F., Green, D. G., & Insa-Cabrera, J. (30 August, 2016a). Factors of collective intelligence: How smart are agent collectives? In *Proceedings of 22nd European Conference on Artificial Intelligence (ECAI)* (pp. 542–550). The Hague, The Netherlands.

Chmait, N., Li, Y.-F., Dowe, D. L., & Green, D. G. (30 August, 2016b). A Dynamic intelligence test framework for evaluating AI agents. In *Proceedings of Evaluating General-Purpose AI (EGPAI), ECAI workshop*. The Hague, The Netherlands.

Churchill, D., & Buro, M. (2012). Incorporating search algorithms into RTS game agents. *Artificial Intelligence in Adversarial Real-Time Games: Papers from the 2012 AIIDE Workshop*, 2–7.

Collins, A. M., Rinderer, T. E., Tucker, K. W., Sylvester, H. A., & Lackett, J. J. (1980). A model of honeybee defensive behaviour. *Journal of Apicultural Research, 19*(4), 224–231. https://doi.org/10.1080/00218839.1980.11100029.

Corkill, D. D. (1991). *Blackboard systems. AI expert, 6*(9), 40–47.

Corkill, D. D. (2003). Collaborating software: Blackboard and multi-agent systems & the future. In *Proceedings of the International Lisp Conference*. New York, NY. http://mas.cs.umass.edu/pub/paper_detail.php/265. Accessed March 22, 2017

Daylamani-Zad, D., Graham, L. B., & Paraskevopoulos, I. T. (2017). Swarm intelligence for autonomous cooperative agents in battles for real-time strategy games. In *Proceedings of the IEEE 9th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)* (pp. 39–46). Athens, Greece, 6–8 September. IEEE. https://doi.org/10.1109/vs-games.2017.8055809

Detrain, C., & Deneubourg, J.-L. (2008). Collective decision-making and foraging patterns in ants and honeybees. In AS. Raikhel (Ed.), *Advances in insect physiology* (p. (35) 123–173). London: Academic Press. https://doi.org/10.1016/s0065-2806(08)00002-7

Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science, 344*(2–3), 243–278. https://doi.org/10.1016/j.tcs.2005.05.020.

Garnier, S., Gautrais, J., & Theraulaz, G. (2007). The biological principles of swarm intelligence. *Swarm Intell, 1*(1), 3–31. https://doi.org/10.1007/s11721-007-0004-y.

Gomes, P., Paiva, A., Martinho, C., & Jhala, A. (2013). Metrics for Character believability in interactive narrative. In H. Koenitz, T. I. Sezen, G. Ferri, M. Haahr, D. Sezen, & G. Çatak (Eds.), *Interactive storytelling* (Vol. 8230 LNCS, pp. 223–228). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-02756-2_27

Hernández-Orallo, J., & Dowe, D. L. (2010). Measuring universal intelligence: Towards an anytime intelligence test. *Artificial Intelligence, 174*(18), 1508–1539. https://doi.org/10.1016/j.artint.2010.09.006.

Hunt, G. J. (2007). Flight and fight: A comparative view of the neurophysiology and genetics of honey bee defensive behavior. *Journal of Insect Physiology, 53*(5), 399–410. https://doi.org/10.1016/j.jinsphys.2007.01.010.

Huo, P., Shiu, S. C. K., Wang, H., & Niu, B. (2009). Application and comparison of particle swarm optimization and genetic algorithm in strategy defense game. In *Proceedings of Fifth International Conference on Natural Computation* (pp. 387–392). Tianjian, China. 14–16 August: IEEE. https://doi.org/10.1109/icnc.2009.552

Insa-Cabrera, J., Hernández-Orallo, J., Dowe, D. L., Espana, S., & Hernández-Lloreda, M. V. (2012). The anYnt project intelligence test: Lambda-one. In *Proceedings of AISB/IACAP 2012 Symposium "Revisiting Turing and his Test: Comprehensiveness, Qualia, and the Real World"* (pp. 20–27). Birmingham, UK, 4–5 July.

Johnson, B. R. (2010). Division of labor in honeybees: Form, function, and proximate mechanisms. *Behavioral Ecology and Sociobiology, 64*(3), 305–316. https://doi.org/10.1007/s00265-009-0874-7.

Karpov, I. V, Johnson, L. M., & Miikkulainen, R. (2015). Evaluating team behaviors constructed with human-guided machine learning. In *Proceedings of the IEEE Conference on Computational Intelligence in Games*. http://nn.cs.utexas.edu/?karpov:cig15

Karpov, I., Johnson, L., Valsalam, V., & Miikkulainen, R. (2012). Evaluation methods for active human-guided neuroevolution in games. In *2012 AAAI Fall Symposium on Robots Learning Interactively from Human Teachers (RLIHT)*. http://nn.cs.utexas.edu/?karpov:aaaifss12

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). Paris, France. 27 November–1 December: IEEE. https://doi.org/10.1109/icnn.1995.488968

Li, S., & Hu, F. (2009). Communication between the RoboCup agents based on the blackboard model and observer pattern. In *Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing* (pp. 1–5). Marrakech, Morroco. 24–26 September: IEEE. https://doi.org/10.1109/wicom.2009.5304011

Lieck, R., & Toussaint, M. (2016). Temporally extended features in model-based reinforcement learning with partial observability. *Neurocomputing, 192,* 49–60. https://doi.org/10.1016/j.neucom.2015.12.107.

Løvlid, R. A., Bruvoll, S., Brathen, K., & Gonzalez, A. (2017). Modeling the behavior of a hierarchy of command agents with context-based reasoning. *The Journal of Defense Modeling and Simulation*. https://doi.org/10.1177/1548512917702832.

Mausam, & Kolobov, A. (2012). Planning with Markov decision processes: An AI Perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning, 6*(1), 1–210. https://doi.org/10.2200/S00426ED1V01Y201206AIM017.

Menzel, R. (1990). Learning, memory, and 'cognition' in honey bees. In R. P. Kesner & D. S. Olton (Eds.), *Neurobiology of comparative cognition* (pp. 237–292). Hillsdale, NJ: Erlbaum Inc.

Miller, K. J., Brody, C. D., & Botvinick, M. M. (2016). Identifying Model-Based and Model-Free Patterns in Behavior on Multi-Step Tasks. *bioRxiv*. http://biorxiv.org/content/early/2016/12/24/096339.abstract

Mock, K. J. (2002). Hierarchical heuristic search techniques for empire-based games. In *Proceedings of the International Conference on Artificial Intelligence (IC-AI)* (pp. 643–648). Las Vegas, Nevada, USA, June 24–27.

Orkin, J. (2011). Using online games to capture, generate, and understand natural language. In *the 13th European Workshop on Natural Language Generation* (p. 71). Nancy, France: Association for Computational Linguistics.

Orkin, J., & Roy, D. (2007). The restaurant game: Learning social behavior and language from thousands of players online. *Journal of Game Development, 3*(1), 39–60.

Orkin, J., & Roy, D. (2009). Automatic learning and generation of social behavior from collective human gameplay. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, *1*, 385–392. http://dl.acm.org/citation.cfm?id=1558013.1558065

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 25–34). New York, USA: ACM. https://doi.org/10.1145/37401.37406

Robertson, G., & Watson, I. (2014). A review of real-time strategy game AI. *AI Magazine, 35*(4), 75–104. https://doi.org/10.1609/aimag.v35i4.2478.

Rogers, K. D., & Skabar, A. A. (2014). A micromanagement task allocation system for real-time strategy games. *IEEE Transactions on Computational Intelligence and AI in Games, 6*(1), 67–77. https://doi.org/10.1109/TCIAIG.2013.2297334.

Russell, S., & Norvig, P. (2009). *Artificial intelligence: A modern approach* (3rd ed.). Upper Saddle River, NJ: Prentice Hall. https://doi.org/10.1017/S0269888900007724.

Scheidt, D. H., & Pekala, M. J. (2007). Model-based agents. In *Power Engineering Society General Meeting, 2007. IEEE* (pp. 1–2). IEEE.

Seeley, T. D., Visscher, P. K., & Passino, K. M. (2006). Group decision making in honey bee swarms. *American Scientist, 94*(3), 220.

Stanescu, M., Barriga, N. A., & Buro, M. (2014). Hierarchical adversarial search applied to real-time strategy games. In *Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (pp. 66–72). Raleigh, NC, USA, October 03–07: AAAI Press. http://dl.acm.org/citation.cfm?id=3014752.3014763

Synnaeve, G., & Bessière, P. (2011). A Bayesian model for RTS units control applied to StarCraft. In *2011 IEEE Conference on Computational Intelligence and Games, CIG 2011* (pp. 190–196). https://doi.org/10.1109/cig.2011.6032006

Tavares, A. R., Azpúrua, H., & Chaimowicz, L. (2014). Evolving swarm intelligence for task allocation in a real time strategy game. *Brazilian Symposium on Games and Digital Entertainment, SBGAMES*, *2014–Decem*(December), 99–108. https://doi.org/10.1109/sbgames.2014.17

Tavares, A. R., Zuin, G. L., Chaimowicz, L., & Azpúrua, H. (2017). Combining genetic algorithm and swarm intelligence for task allocation in a real time strategy game. *Journal of Interactive Systems*, *8*(1), 4–19. http://seer.ufrgs.br/index.php/jis/article/view/56146/43683

von Frisch, K. (1965). *Tanzsprache und Orientierung der Bienen*. Berlin: Springer.

Weber, B. G., Mawhorter, P., Mateas, M., & Jhala, A. (2010). Reactive planning idioms for multi-scale game AI. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, CIG2010* (pp. 115–122). https://doi.org/10.1109/itw.2010.5593363

Yannakakis, G. N., & Togelius, J. (2015). A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games, 7*(4), 317–335. https://doi.org/10.1109/TCIAIG.2014.2339221.

Zhang, S., Bock, F., Si, A., Tautz, J., & Srinivasan, M. V. (2005). Visual working memory in decision making by honey bees. *Proceedings of the National academy of Sciences of the United States of America, 102*(14), 5250–5255. https://doi.org/10.1073/pnas.0501440102.

Damon Daylamani-Zad is a Senior Lecturer in Games and Computing in the Department of Computing and Information Systems at University of Greenwich. He is a Fellow of the British Computing Society and holds a BSc in Software Engineering from University of Tehran, an MSc in Multimedia Computing and a PhD in Electronic and Computer Engineering both from Brunel University London where he has also been an EPSRC Research Fellow. Damon's research interests focus on Digital Games, including Collaborative Games, Serious Gaming, and Player Modelling and Personalisation especially in MMOGs (Massively Multiplayer Online Games) as well as application of Artificial Intelligence in Games and Digital Media, and Collaborative Content Modelling, the application of MPEG standards. He has published his research findings widely in journals, edited books and presented his work at several conferences including those hosted by the IEEE.

Letitia B. Graham is a developer at Bossa Studios, London, UK. She holds a BSc (Hons) in Computing with Games Development from University of Greenwich and has collaborated with various games and digital media companies including Bossa and Disney. Letitia is interested in application of Artificial Intelligence in Games and Digital Media, and multi-agent architectures. She has published her research findings widely and presented her work at several conferences including those hosted by the IEEE.

Ioannis Paraskevopoulos is a technical leader and project manager for immersive technologies, as well as IoT, desktop, web and mobile projects in the Altran Italia's Software Engineering Expertise Center. He is also a software architect and engineering consultant with expertise in and focus on design and implementation of VR, AR, MxR, IoT and mobile applications. He was previously a Senior Lecturer in Games and Computing in the Department of Computing and Information Systems at University of Greenwich. He is a Fellow of the British Computing Society and holds a PhD in Electronic and Computer Engineering from Brunel University London. He has been the receiver of an industrial secondment from Royal Academy of Engineering. He has published his research findings widely in journals, edited books and presented his work at several conferences including those hosted by the IEEE.