A Framework for Implementing Deliberative Agents in Computer Games

N.P.Davies, Q.H.Mehdi and N.Gough

Computer Games Centre
School of Computing and Information Technology
University of Wolverhampton
Wolverhampton, UK
E-mail: N.P.Davies2@wlv.ac.uk

KEYWORDS

Game: Deliberative Agents, Computer Games, Artificial Intelligence.

ABSTRACT

With the emergence of complex computer games and advanced gaming hardware, possibilities for overcoming some of the deficiencies in traditional game AI are becoming feasible. These deficiencies (repetitive, predictable, inhuman behaviour) are caused by the reliance on simple, reactive AI techniques for in game characters, and can be overcome by using more sophisticated AI and agent techniques. The aim of our research is to create new forms of intelligent characters (agents) that will exhibit human-like intelligence and provide more challenging and entertaining virtual opponents and team mates for computer games. We present here an overview of the Belief Desire Intention (BDI) model of agency, and its applicability to computer games, and present our prototype application that implements a BDI agent system within the 3D computer game Unreal Tournament via GameBots and JavaBots technology. We also outline our future goals for improving the system via the introduction of multi agent scenarios that require cooperation between self-interested autonomous entities.

INTRODUCTION

With this continued popularity of computer games, game players are expecting new challenges with more sophisticated games and game playing experiences. Increases in processing power are now giving game developers opportunities to develop novel techniques to incorporate into their games. With graphics capabilities now reaching the point where game environments are becoming almost photorealistic, as seen in games such as Half Life 2 (Valve software, 2004), some of this power must become available for AI systems. Currently developers are looking for new and inventive ways to keep the game players entertained. One way games have evolved over the last few years is with the introduction of on-line play, where players can compete against other players from across the world. This has been developed at great expense and risk to game developers (Hunt, 2003). One reason players are turning towards online play is that they can identify problems with current artificial agents in computer games, and prefer to play against 'intelligent' opponents. Competent gamers can now spot the 'cheats' game developers use to simulate intelligent looking behaviour within AI 'bots'. This can lead to a breakdown in the players' immersion within an online world (Welsh 2005). It is unclear whether game players inherently dislike playing against artificial opponents, or whether they dislike playing against unintelligent opponents; however, whether intelligence is real, or simulated, when the illusion of intelligence is removed players can feel let down by the experience. It is anticipated that artificial intelligence will continue to be an important factor in the next generation of computer games. The challenge is to produce artificial intelligence for computer games characters that can utilise the increased power afforded by improvements in games hardware, and make AI agents appear as human-like as possible so as to improve the game playing experience.

Developing better AI for computer games is not a simple task, as the game environments tend to be 'open' (dynamic, continuous, nondeterministic), which are the most difficult environments for designers to construct agents for (Russell & Norvig, 1995). The agents that appear in game worlds need to be reactive, autonomous, and goal-oriented if they are to survive; another complex design problem. To produce agents capable of this behaviour in these environments we propose using the Belief-Desire-Intention (BDI) model of agency (Bratman, 1987). In this model, agents are constructed using humanistic concepts such as goals to achieve, beliefs about the environment, and plans to achieve goals, based on beliefs. Using BDI, we can simulate the decision making processes performed by humans, using the same information available to humans, in order to make the agent act in a human like way. It is expected that this will make computer game characters appear more realistic, and therefore, improve the experience of playing against artificial game characters. The paper is constructed as follows. In next section, we outline the philosophical roots of BDI, including its folk psychology and practical reasoning background. We then outline the design of our system for integrating a BDI reasoning engine into the computer game Unreal Tournament, and detail the three layers of the system including

Proceedings 20th European Conference on Modelling and Simulation
Wolfgang Borutzky, Alessandra Orsoni, Richard Zobel © ECMS, 2006
ISBN 0-9553018-0-7 / ISBN 0-9553018-1-5 (CD)

deliberation, communication, and virtual environment. We follow this with our experimental results, including the initial implementation where agents interact with the game environment, exploring, attacking enemies, producing paths through the environment, following path, and building health by locating health packs. We conclude with our future aims, including the addition of a multi-agent layer for common goals amongst agents.

BACKGROUND

The reasoning system used by agents in our architecture is based on the Belief Desire Intention model of agency, as postulated by Michael Bratman (1987). This in turn is based on folk psychology, and has its foundations in the discipline of practical reasoning. According to Stich and Ravenscroft (1994), folk psychology has evolved out of the common sense expressions people use to describe the behaviour of themselves and others in terms of mental concepts such as beliefs, desires, hopes and fears. It is not a theory formulated by experimental research, and as such, has some detractors as to its philosophical worth, Sellars (1956). However, folk psychology does have its uses, and gives developers a simple tool for describing the actions of others. Using folk psychology descriptions, an agent can be developed with simple to understand statements e.g. if health is low, get health pack. It can also use the description to maintain an understanding of its environment, itself, and the other entities. As there may be many conflicting options open to an agent simultaneously, the notions of practical and means-end reasoning are introduced to drive action. Practical reasoning is the process of weighing up all the alternative options available to the agent that are consistent with its beliefs and desires. Means-end reasoning identifies how to achieve these goals by using the abilities available to it. Practical reasoning agents are rational; that is, they only adopt goals they believe are achievable and logically consistent. For example, an agent 'A' wishes to be in state 'X', and believes that by doing action 'Y', it will be in state 'X'; therefore it does 'Y'. An agent will not act irrationally. It will not attempt to do something it does not believe it can do.

To apply folk psychology principles to a computational device requires that the ideas are formulated mathematically, and reduced in complexity. To this end, BDI architecture becomes a good candidate. In BDI, these concepts are reduced to two main functions; beliefs about the environment, and the desires to be in certain states in that environment. Bratman also introduced the concept of intention whereby agents commit themselves to achieving certain goals, and will try every method at their disposal to achieve the intention, until it becomes impractical / illogical to do so. This commitment differs from mere desires, which can be considered ideal aims. By using intentions, the agent can discard information that is inconsistent or irrelevant to the adopted goals, and the processing requirement is therefore much reduced. The foundations of BDI have been formulated into a set of logics by groups including Rao and Georgeff (1995), and implemented in a number of successful systems, including, JACK(Agent Oriented Software, 2006), and Jadex(Braubach et al, 2006). With the development of these tools, it proves the feasibility of developing agents that use humanistic concepts on computer systems.

We are currently implementing a BDI based agent architecture for agents as opponents in First Person Shooter (FPS) games. AI in FPS games has been evolving for over a decade. In the first increments of the games, such as Quake (1996), the AI was very simple. Opponents simply stayed in one location in the game, and ran towards opponents shooting their weapons when they came into sight. Different skill levels were achieved by making AI more accurate, increasing the damage caused by weapons, or by simply adding more enemies to games. Later games achieved more realistic behaviour by introducing state machines that allowed agents to exhibit a wider range of behaviours, such as wandering, attacking, exploring etc. Concepts such as path finding were also introduced that enabled characters to navigate the map. AI has continued in its progression to current levels, which has become very sophisticated. Valdes (2004) indicates how far AI has come with particular reference to the game Halo2, where agents are beginning to reason about the gaming environment based on perceptual input. However, sophisticated as this AI is, it is still based upon state machines, path finding, and scripted behaviour. Welsh (2005) points out that human players are very good at identifying anomalies in systems such as these. In our system, we are modelling the thought processes game players use. It is expected that in this way we can overcome some of the deficiencies created by the reliance on state machines and scripting, to produce agents that appear more human-like.

SYSTEM DESIGN

Our framework, (Figure 1), is constructed using three layers that integrate several tools available as open source projects. The layers are:

Intelligent Agent	:	Jadex
Game Environment	:	Unreal Tournament
Communication Layer	:	JavaBots/GameBots

Each layer is described in more detail below. It should be noted that our implementation is client-server based, and as such, requires an extra layer for external communication with the game engine. This means the agent is a combination of two separate entities. The first entity is situated within the Unreal Tournament game, and can be considered an avatar for our intelligent agent. The intelligent agent guides the avatar by sending commands over the network, and builds up a view of the environment by receiving perception messages. There are several reasons the system is implemented in this way;

not least is the desire to implement our system in a modern commercial computer game. It is not possible to incorporate our AI directly within the game due to limitations in access to the engines source code. However, this architecture has the benefit of forcing the agent to play the game in the same way as human players, based on sensor information. In addition, this type of architecture allows experimentation with human / agent teams in an extensible framework, and allows many agents to connect to the game server simultaneously.

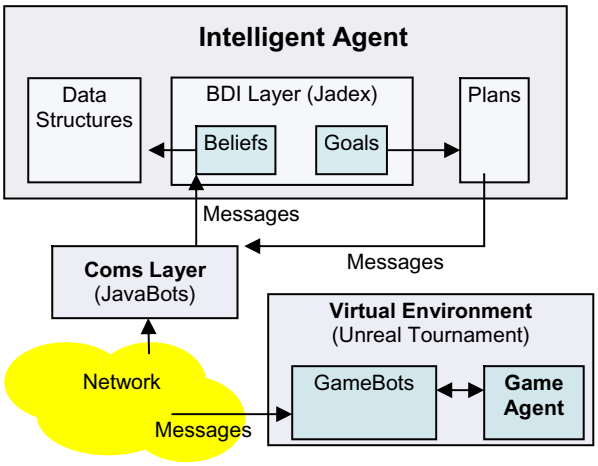

Figure 1: System Design

Intelligent Agent

The intelligent agent layer consists of a reasoning system, a knowledge base of plans, and data structures for storing environment information. The reasoning system is developed using Jadex (Braubach et al, 2004). This is an agent platform that allows the creation of BDI agents in the Java programming environment i.e. it allows the creation of agents that use the mental attitudes of belief, desire, and intention to model human like reasoning processes. Agents are created via the specification of beliefs, goals, plans, events, and capabilities. Goals relate to the state an agent would like to achieve and can be of several types; achieve, maintenance, and perform. Achieve goals are used to perform an action, such as move to a waypoint. This type of goal can either succeed or fail. Maintain goals are used to monitor the agent, e.g. make sure health stays above 50. If the agent's health drops below this level, then some action is triggered to rectify the situation. Perform goals are used to perform actions that are consistent with a state that don't have a target state to reach e.g. 'explore' where an agent will continue to search an environment for as long as it wishes. To accomplish goals, relevant plans are used. Plans are created via extending an abstract Plan class that allows messages to be sent between plans and agents. In the ADF, a plan's applicability to a goal is specified with a trigger statement. There may be many plans applicable to specific goals, therefore the plans applicability can be further reduced with clarifiers such as context conditions, which states that plans can only be created if certain belief conditions are met. Beliefs specify agents' knowledge of the world, and are used to trigger goals, and plans success or failure. Beliefs can be any Java object. In our system, we have created objects to store an agent's status, enemy locations, navigation info etc. Using this system, we have developed AI agents that have the ability to respond to environmental events, identify appropriate plans to handle the events, and execute those plans in a timely manner. While executing plans, the agents monitor the environment, and internal belief structures, in order to ensure plans are still relevant.

Virtual Environment

Our intelligent agent connects to a game environment, and interacts with it through perceptions and performing actions. We have chosen to develop our system using the game engine Unreal Tournament (Figure 2); a three-dimensional, networked FPS computer game. The game includes several game types. These include Death Match; a free-for-all match with the winner decided by the highest number of frags achieved over a certain time period, Domination; where players compete to capture and defend domination points for a specified amount of time, Capture the flag; where players have to retrieve the opposition's flag and bring it back to their base, while defending their own flag and Team Death Match; where teams work together to achieve the highest frag rate in a set time period. The game can be modified in several ways via an editor and a scripting language. New levels can be created in the graphical designer Unreal Edit. Game rules and physics can be modified via the scripting language Unreal Script. Lewis and Jacobson (2002) point out the benefits of this solution. The engine is inexpensive. The graphics rendering capability is superior to anything that can feasibly be created by small research groups. Also, the game logic is fully implemented i.e. the game comes complete with standard functions such as collision detection, physics systems, game maintenance etc. Other benefits include the large user base of players of the game. This gives access to domain experts for knowledge elicitation, and groups for evaluation of the completed system. The use of a game engine is not an ideal solution however. It would be better to develop a complete computer game where all functions are accessible at a source code level; a level that is unavailable through the use of game engines. Game engines require us to work within the limits imposed by the game engine developers, which can cause problems. Norling (2004) carried out work related to our research using the Quake engine. A number of limitations were reported where certain behaviours could not be modelled due to the deficiencies in information sent from the game server. However, in the short term, Unreal Tournament offers a neat solution to our visualisation and game environment requirements.

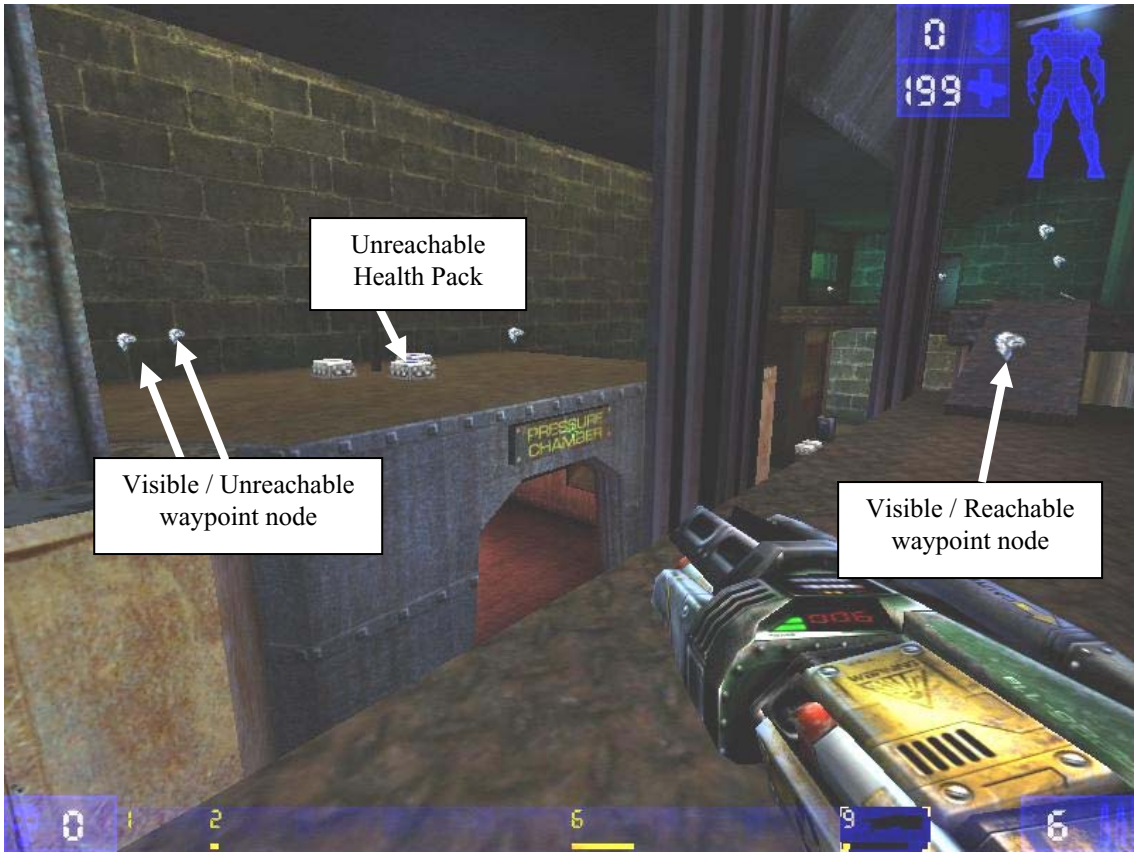

Figure 2: Screen shot of Unreal Tournament

Communication Layer

To facilitate communication between the intelligent agent and the game world we have adopted the use of the dual middleware product of GameBots/JavaBots (Marshall et al, 2006). GameBots is an extension to Unreal Tournament that resides upon the game server. It is written in Unreal Script and extends the basic AI and networking components shipped with the game. GameBots allows external processes to access internal AI functions through a network socket connection. For every game loop, agent perception data is sent as a synchronous message packet across the network to a client. This information consists of currently visible navigation points, inventory items, and other visible agents. Status information is also sent including the agent's health, location, weapons etc. Event messages, such collision information, damage reports etc, are sent when they occur in the game via asynchronous messages. The communication is a two way process, and GameBots can receive messages that consist of actions for the agent to perform. Actions include rotate, walk, run, shoot etc, and also more complex operations such as find path, which queries the navigation system of Unreal Tournament, and sends a path list back in the form of an asynchronous message. The client portion used by the intelligent agent is called JavaBots; a Java based system developed to connect to GameBots. It is an extensible API that contains example bots, visualisation applications, and a Bot Runner application that allows agents to be connected and visualised via a GUI interface. We have taken the original JavaBots project and removed the extra functionality to produce a very simple API that simply listens for messages, and sends instructions back to Unreal Tournament. We have removed the sample bots and Bot Runner classes, and instead use functions within Jadex to maintain the connection to the game.

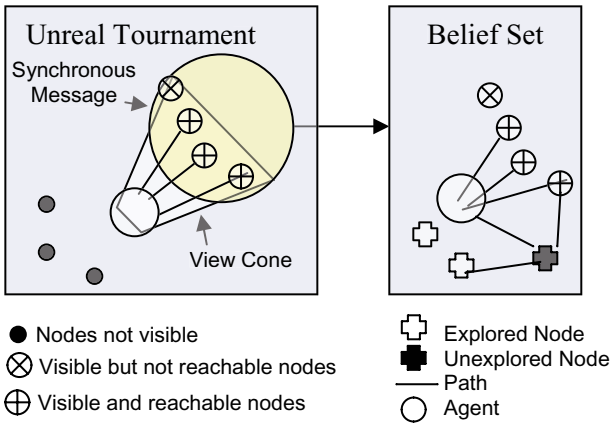

Figure 3: Navigation messaging system

Figure 3 shows an example of the messaging system in which GameBots sends a message block containing navigation point information to the intelligent agent, where it is recorded in a belief structure. In Unreal Tournament, the game agent has a set view cone of around 45 degrees. At any point in the game, the game agent is capable of observing a discrete portion of the game environment contained within this view cone, which is not occluded by walls. Figure 2 shows a scene from Unreal Tournament to indicate a typical view. In the view, and game agent can see inventory items (Heath Packs), and waypoint nodes. The waypoint nodes can either be reachable, or unreachable. In the illustration, there is a gap between the game agent, and a platform containing the health packs; it can therefore see them, but cannot reach them directly. The navigation node to the right of the illustration is visible and reachable directly. Therefore, the agent can run directly to it. This information is grouped into a single message block, and sent to the intelligent BDI agent. The intelligent agent receives this message, parses it, and populates its belief sets. New nodes (nodes an agent has not seen before) are added to the list of known nodes. Nodes at the position of the agent are marked as 'visited' to indicate the agent has explored the position. Two other data sets are populated; visible nodes and reachable nodes. At each frame, these sets are cleared, and populated with the new data contained in the message i.e. only nodes that the agent can currently see are stored. All historical data is deleted.

Visualisation Layer

In addition to the single agent messaging system, GameBots also has a useful feature that exports the complete game state connection. This information includes the location of all objects, agents and waypoints currently in the game. It is not intended for agents to use this information to influence behaviour, as this would give the agents an unfair advantage by giving it extra perceptions unavailable to human players. However, it is a useful method for developers to view the current game state for evaluation purposes. We are currently expanding the Java visualisation application to incorporate Java3D functionality. This will give us the ability to view the game in 3 dimensions, and gain a better understanding of what is taking place within a game session.

EXPERIMENTAL RESULTS

A prototype application has been developed that shows the potential of the architecture. The intelligent agents are able to connect to Unreal Tournament, build a 3D view of the environment, and navigate the world. The agents are also capable of some basic behaviour in the game. This includes exploring, navigating, hunting and escaping. The overriding maintenance condition of the agent is to maintain health above 50. Once health drops below this level, the agent attempts to disengage from a combat situation, and find health until its health has reached 90 or above. This behaviour is achieved via a maintenance goal that inhibits the 'explore' and 'attack' goals. When the agent is attempting to build its health, it will observe the environment to see if any health packs are currently visible and reachable. If it can see a health pack, it will move towards it and pick it up. If there are no visible health packs, it checks its memory to recall the location of the nearest health pack to its current location. At this point, it queries Unreal Tournament to find a path to it and then follow this path to the health pack. If, on route, it spots another health pack that it had not previously seen, it will temporarily drop the goal of following the path in favour of collecting the new health pack. It will then resume the goal of following the path (assuming it still requires health). If the agent encounters an enemy agent while following the path, it will retreat, and choose a path to an alternate health pack. This behaviour is created using achieve goals. At each level, the agent can either succeed or fail. If a section of the plan fails, the agent is capable of retrying the goal, or dropping the goal and starting again at any level. If the agent is not finding health packs, the agent will then explore the environment. It will view its reachable nodes list, check if any of them have not been explored before and if it finds an unexplored node, it will move towards it. If it cannot see an unexplored node, it will run to a random reachable node. This behaviour is created using a perform goal, and the agent will continue with this behaviour until some event causes it to drop the behaviour. An example of a condition where the agent will drop the goal is if the agent spots an enemy. When an enemy is spotted, the agent will engage in an attacking behaviour that includes firing its weapon, running towards the enemy, and jumping left and right until it has either killed the enemy, its health drops to a point where the maintenance condition forces it to try to escape, or it is killed. The behaviour of the agent is presently for illustrative purposes, and is not intended as a sophisticated behaviour system. However, it does prove that developing a more complex agent is possible with a combination of goals and plans within the BDI framework.

CONCLUSIONS AND FUTURE WORK

It has been proposed that the goal based, deliberative architecture, BDI, has the potential to produce more human like behaviour in computer game characters, which will, therefore, exhibit more realistic behaviour than can be achieved with simple reactive AI techniques alone. We have identified a set of tools, and implemented a prototype application that incorporates the commercial computer game Unreal Tournament, the reasoning layer BDI through Jadex, and linked the systems using the communication system GameBots/JavaBots. We have created a set of agents that perceive their environment via sensory information gathered from the game, and use this information to build up a beliefs base. Using these beliefs, and a set of desired conditions, the agent uses its plan base to bring about beneficial states of affairs. The implementation currently allows the agent to explore the

environment, attack enemies, find health packs, and navigate the game map by following a path list. In the next stage of development, we will expand upon this basic bot, and add more sophisticated behaviour and tactics. We will also implement team based behaviours, which will require agent negotiation techniques and social abilities. In addition, we will expand upon the BDI framework, and incorporate a layered architecture where fast, reactive behaviour can be accomplished in lower levels, and high level cooperative goals can be accomplished at higher levels.

REFERENCES

Agent Oriented Software Pty. Ltd. JACK Intelligent Agents. (2006) http://www.agent-software.com

Bratman, M. (1987) *"Intention, Plans, and Practical Reason"* Harvard University Press: Cambridge, USA

Braubach, L., Pokahr, A., Lamersdorf, W., Krempels, K., Woelk, P. (2004) *"A Generic Simulation Service for Distributed Multi-Agent Systems"*, in: From Agent Theory to Agent Implementation (AT2A1-4) 2004

Epic Games Inc. Unreal Tournament. http://www.unreal.com

Hunt, B. (2003) 'Live and kicking", Financial Times Online,http://www.vmc.com/articles/Financial%20Times%20Article.pdf

Lewis, M., Jacobson, J., (2002), *'Game Engines in Scientific Research'*, Communications of the ACM, Volume 45, Issue 1. New York, USA

Marshall, A. N., Gamard, S., Kaminka, G. J., Manojlovich, Tejada S., Gamebots, viewed 10th Mar 2006, http://planetunreal.com/gamebots/

Norling, E. (2004) *'Folk psychology for human modelling: extending the BDI paradigm'*. In : Int. Conf. on Autonomous Agents and Multi Agent Systems (AAMAS), New York, 2004.

Rao, A., Georgeff, M., (1995) *'BDI agents: from theory to practice'* Tech. Rep. 56, Australian Artificial Intelligence Institute, Melbourne, Australia

Russell, S., Norvig, P., Artificial Intelligence: A Modern Approach. Prentice-Hall, Englewood Cliffs, New Jersey, 1995.

Sellars, W. (1956). ''Empiricism and the Philosophy of mind,'' in H. Feigl & M. Scriven, eds., *Minnesota Studies in the Philosophy of Science*, 1, Minneapolis: University of Minnesota Press.

Stich, S. & Ravenscroft, I. (1994): "What is Folk Psychology?". *Cognition* 50: 447-68

Valve Software (2004) Half Life 2, http://www.half-life.com/

Valdes, R.: In the Mind of the Enemy: The Artificial Intelligence of Halo 2 (2004): http://stuffo.howstuffworks.com/halo2-ai.htm

Welsh, T., (2005) *'A Typology of Givaways: An Evaluation of FPS Bots'* in Proceedings of CGAIMS'2005 6th International Conference on Computer Games: Artificial Intelligence and Mobile Systems 27-30 July 2005

AUTHOR BIOGRAPHIES

NICHOLAS P. DAVIES was born in Wolverhampton, UK, where he studied Computer Science at the University of Wolverhampton, and obtained a First Class Degree in 2003. He is currently researching AI and Computer Games, and has completed the first two years of his PhD.