Mainstream Games in the Multi-Agent Classroom

Celso de Melo
cmme@mega.ist.utl.pt

Rui Prada
rui.prada@tagus.ist.utl.pt

Guilherme Raimundo
guilherme.raimundo@tagus.ist.utl.pt

Joana Paulo Pardal
joana.paulo@l2f.inesc-id.pt

Helena Sofia Pinto
sofia@vinci.inesc-id.pt

Ana Paiva
ana.paiva@inesc-id.pt

Department of Computer Engineering, IST-Technical University of Lisbon and INESC-ID
Av. Prof. Cavaco Silva – Taguspark
2744-016 Porto Salvo
Portugal

Abstract

Computer games make learning fun and support learning through doing. Edutainment software tries to capitalize on this however, it has failed in reaching the levels of motivation and engagement seen in mainstream games. In this context, we have integrated a mainstream first-person shooter game, Counter-Strike, into the curriculum of our Autonomous Agents and Multi-agent Systems course. In this paper we describe this integration and a platform to support the creation of Counter-Strike agents. In addition, a questionnaire was posed to our students to assess the success of our approach. Results show that students found the idea of applying a first-person-shooter game motivating and the integration with the curriculum useful for their education.

Keywords: Teaching Multi-Agent Systems; Mainstream Games; Multi-Agent Platforms

1 Introduction

Computer games are, evermore, part of our culture. The global market is worth billions of dollars [1]. The average game player age is, now, around thirty years old and both genders are spending considerable time playing games. Why people are motivated to play games is still subject of discussion but, commonly accepted reasons are fantasy, challenge and curiosity [2]. The industry has acknowledged this and is spending a lot of time and money producing content to meet these needs. However, besides entertainment, computer games support skill development such as logical thinking and problem solving skills, as well as collaborative learning [3].

This paper describes the integration of a first-person-shooter mainstream game with the curriculum of the *Autonomous Agents and Multi-agent Systems* one-semester course in the 4th year of the Informatics and Computer Engineering undergraduate degree at the IST-Technical University of Lisbon. This integration relies on a platform developed throughout the past two years, which supports the creation of agents that sense and act on the world, as well as, communicate with each other. Using this platform, students were asked to explore different architectures, cooperative behavior and one additional topic (e.g., learning or personality). Finally, a tournament took place to confront students' agents.

The paper is organized as follows. Section 2 describes relevant background. Section 3 describes the chosen first-person-shooter game. Section 4 describes the multi-agent platform. Section 5 describes the integration of the platform with the course. Section 6 presents the results of a study conducted to evaluate our approach. Finally, section 7 draws some conclusions.

2 Background

Development of games for education, or *edutainment* software, is based on two key aspects: (a) making learning fun; (b) learning through doing. However, edutainment has failed to reach the levels of engagement seen in *mainstream games* because: (a) it tends to be too simplistic; (b) the tasks are repetitive; (c) the target audience becomes aware that it is being coerced into 'learning'. Thus, mainstream games emerge as an alternative to edutainment software. Forced by an extremely competitive market, extensive investment is placed on creating compelling storylines and audiovisual experiences for these games. [4]

In this sense, this work describes the integration of a mainstream game with a multi-agent systems course. Our approach contrasts with several multi-agent platforms which relate to the edutainment approach. Here, two kinds of platforms can be discerned. In the first case, they tend to have simplified programming languages, good documentation and libraries with predefined templates. A representative platform is NetLogo [5] which embeds a high-level programming language in an integrated, interactive modeling environment. It comes with a models library with several simulations ready to be explored, supports collaborative exploration through a client/server architecture and has been used in K-12, high-school and undergraduate courses. In the second case, the platforms define or integrate with a specific multi-agent systems game. Representative is e-Game [6] which allows students to explore several auctions and negotiation protocols in an electronic market context and is based on the Trading Agent Competition [7]. However, in general, these educational platforms suffer from the same kind of problems seen in edutainment software. Effectively, simplified worlds and poor visualization capabilities fail to motivate students in the way a mainstream game does. Still, they are good tools to introduce multi-agent concepts and, thus, we use NetLogo in the first weeks to explain basic multi-agent concepts. However, for the remainder of the semester, students apply and explore these concepts, in the more challenging and motivating game world.

A different set of platforms provides specialized libraries usually for particular domains. A representative platform is JADE [8] which is a FIPA-compliant multi-agent framework with good support for agent mobility and a broad library of communication protocols. Though not originally conceived for educational purposes, these platforms can be applied in the classroom. In concrete, specialized libraries available with these platforms can be integrated with the respective learning module. Thus, in our approach, we use JADE, in the first weeks, to introduce several communication and cooperation protocols. However, as in education-oriented platforms, by themselves these platforms fail to provide engaging contexts as mainstream games do.

Closer to our approach, GameBots [9] is a multi-agent platform which integrates with a first-person-shooter mainstream game. The platform is based on the client/server architecture, where a central server runs the game and several clients – the agents – connect to it. Sensors and actuators are provided to interact with the world and other agents, as well as tools for visualization and logging of simulation state. Contrasting to this work's education focus, GameBots tries to appeal to the artificial intelligence community introducing the platform as a general test-bed for several research issues. Still, the platform seems to have been applied in educational settings [10] but, no details are provided on integration with course curricula which is as important as the quality of the platform itself.

3 A First-Person-Shooter World

Our work integrates Counter-Strike (CS) [11], version 1.6, in a multi-agent systems course. CS belongs to the first-person-shooter genre where characters confront each other and the player assumes the character's point of view. Furthermore, CS is a *team-based* first-person-shooter, as there are two teams – terrorists and counter-terrorists – which confront each other in several rounds trying to meet some objective or eliminate the opposing team. There are several kinds of objectives, but this work explores only *bomb defusing* maps where terrorists are required to plant and detonate a bomb while counter-terrorists try to prevent them.

Counter-Strike is appropriate for integration with a multi-agent course for several reasons:

(1) *Game play has suitable characteristics for exploration of several multi-agent concepts.* First, cooperation is essential for success in this game, as evidenced by the opinions of professional gamers [12] (see Figure 1). Second, several agent architectures can be explored in the CS dynamic, yet complex, world. Finally, several additional topics can be explored (see section 5);

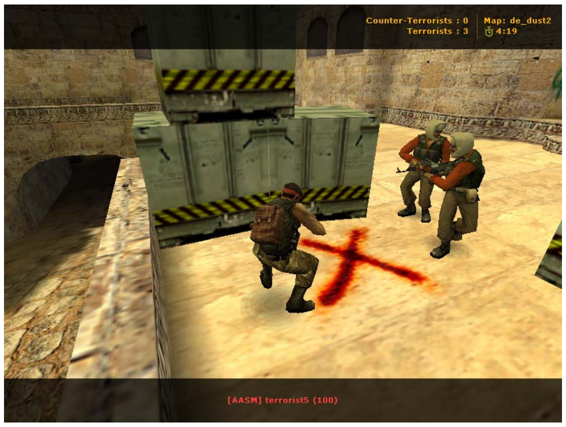

Figure 1. Teamwork is crucial in Counter-Strike. Here, a terrorist plants the bomb while its teammates provide cover. This snapshot is from a student's work.

(2) *It is successful in engaging and motivating players.* First, CS provides a complex world where the outcome is influenced by several factors such as teamwork, exploration strategies and proper use of resources. Second, at the time of writing, CS is the most popular online game [13]. Finally, in professional tournaments, representative of which is Cyberathlete Professional League [12], CS always has the best prizes and greatest number of participants;

(3) *It has a large and active developer community* [14] which continuously improves game play and engine documentation, as well as creates new maps and game modifications.

4 The Multi-Agent Platform

A multi-agent platform for the creation of *bots*, or computer controlled players, in CS was developed. The platform was particularly influenced by HPB bot [15] and

YapB bot [16]. Source code for both is freely available. HPB bot solves the problem of adding bots to CS and proposes graph-based navigation. YapB bot, which expands on HPB bot, defines sophisticated behavior thus, constituting a useful reference on how to interface with the graphics engine which is still poorly documented. However, direct application of either to our course would have been difficult for several reasons: (a) they were not developed with educational purposes in mind; (b) they're poorly documented; (c) the code is poorly organized as it doesn't clearly separate sensors, actuators and decision logic. Thus, for this work, a new platform had to be built.

The platform is divided into several modules: navigation, vision, combat, finances and communication. The navigation module supports graph-based and free navigation. Graph-based navigation relies on files, one per map, which contain the following information: locations in the map; connections between locations; connections properties, for instance, whether a jump is required; important locations such as camping and objective locations. A set of primitives is available to find paths between locations. Two path-finding algorithms are supported: Floyd-Warshall shortest path and A* search. Free navigation supports arbitrary motion in the world and, though harder to control, is important, for instance, to pickup items in the world which are not placed on top of a predefined location. This module also supports simple collision detection and handling. The vision module perceives other agents within a 90º field-of-view. The combat module supports weapon selection, aiming, firing, reloading, bomb planting and defusing. The finances module supports money management. Money is given to players according to their performance in the game and is used to upgrade weapons and armory. Finally, the communication module supports two kinds of communication: radio, corresponding to audible predefined messages; chat, corresponding to string-based communication which supports flexible communication protocols. In both cases, besides message content, information regarding sender and time of emission is sent. Radio messages are broadcast to all teammates while chat messages to teammates or to all players.

The platform supports up to 32 simultaneous players. Players may be agents or humans. Agents interact with the world and other agents exclusively through the aforementioned modules. Every agent's decision cycle is invoked once per frame, though it is not required to produce an action every cycle. Furthermore, every agent executes an independent decision cycle. Thus, for instance, some agents may execute a reactive decision cycle while others a more deliberative one. Together with interaction with human players, this feature supports exploration of complex multi-agent scenarios.

The platform is fully implemented in the general-purpose C++ programming language. A general-purpose language is appropriate for computer engineering students as it provides them with power and flexibility to express their ideas. An additional benefit is that general-purpose languages tend to have wide tool support such as Integrated Development Environments. In our case, MS Visual Studio 2005 was used.

Regarding debugging, the platform supports three options. First, it supports step-by-step debugging, during game play, using a C++ debugger. In this work the MS Visual Studio 2005 debugger was used. Second, it supports printing to a console which is provided by CS. Third, it supports printing to a file which is convenient to log large amounts of information.

Finally, the platform supports a tournament mode. Here, two kinds of agents are confronted in batch-mode. For instance, reactive agents could confront deliberative agents. Tournament setup includes definition of number of rounds, number of agents per team, and scoring policy. The number of rounds should be even as agents assume, in the first half, either the role of terrorists or counter-terrorist and, in the second, the other role. Score is affected by agent eliminations and objective completion, which are automatically registered throughout the game. In the end, both overall and partial results per round are saved in textual and XML format. The latter is useful for automatic processing of data. Tournament mode is useful to compare different kinds of agents, as well as to obtain concrete performance measures.

5 Integration with the Curriculum

The CS platform was integrated with the *Autonomous Agents and Multi-Agent Systems* course in the Computer Engineering undergraduate degree at Technical University of Lisbon. The course curriculum is divided in two blocks (see Figure 2). In the first, spanning the first six weeks, agent architectures, communication and cooperation are introduced. Theoretical background follows Wooldridge's book [17] and NetLogo and JADE are used as supporting tools. In the second block, spanning the next eight weeks, students, in groups of three, apply and explore the aforementioned concepts in CS using the proposed platform. Only bomb defusing maps are considered. Evaluation consists of the CS project (60%) and a written exam (40%).

In concrete, in the second block students are expected to meet the following objectives:

- *Explore several agent architectures* including reactive and Belief-Desire-Intention (BDI) [18]. Due to its simplicity, the reactive architecture is a good place to start as students need time to get acquainted with the platform's modules. Deliberative or hybrid architectures should follow;
- *Model communication and cooperation.* Here, students develop team-based strategies exploring

cooperation techniques and communication protocols learnt in the previous weeks;

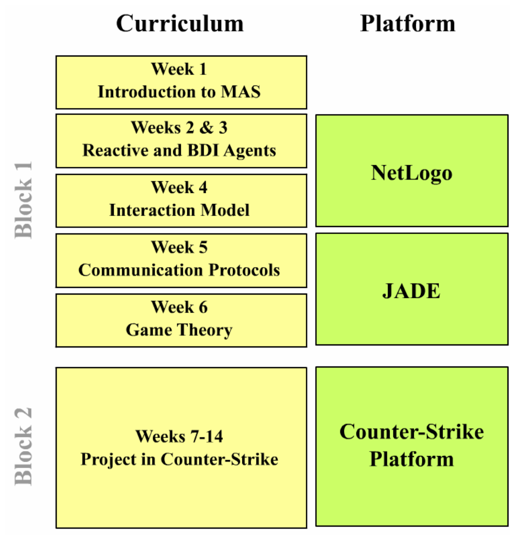
Curriculum
Platform
Week 1 Introduction to MAS
Weeks 2 & 3 Reactive and BDI Agents
Week 4 Interaction Model
Week 5 Communication Protocols
Week 6 Game Theory
NetLogo
JADE
Block 1
Block 2
Weeks 7-14 Project in Counter-Strike
Counter-Strike Platform

Figure 2. The multi-agent course curriculum.

- *Evaluate and compare agent architectures.* Here, students confront developed architectures and determine things like which team is better within a certain architecture or best overall;
- *Explore an additional multi-agent related topic.* Here, the idea is to breed creativity. Topics explored by students included: learning, where, for instance, information about "successful" map locations is learnt by agents; interaction with human players, where cooperation with human players is explored; personality, where agents act according to certain personality traits such as being courageous or scared;
- *Submit one agent architecture to a tournament.* The point of having a tournament is to breed competitive spirit and motivate students. In the end, the best four teams are given a bonus in their grades.

6 Results

In order to evaluate our approach we conducted a questionnaire among the students. We set forth to answer the following questions:

- *Is CS appropriate for our course?*
- *How was CS integration with the curriculum?*
- *How was the CS platform perceived?*

The questionnaire was posed to 79 of our students, corresponding to 66.4% of total enrolled students.

6.1 Is CS appropriate for our course?

To understand whether CS is appropriate for our course students were asked to classify between 1 (totally disagree) and 4 (totally agree) whether they agreed with the following statements: (1) I enjoyed making this project; (2) I think CS is appropriate for this course; (3) I've played CS before.

Results are shown in Table 1. As can be seen, most students (about 83%) agree that CS is appropriate and most students (about 80%) enjoyed the project even though only about 48% knew the game.

Table 1. Results relating the appropriateness of CS for our course. Rating values are 1 (totally disagree); 2 (disagree); 3 (agree); 4 (totally agree). Values represent the number of answers. The last column shows the number of blank answers.

	1	2	3	4	N/A
I enjoyed the project	3	9	46	18	3
I think CS is appropriate for this course	1	7	48	18	5
I've played CS before	31	9	10	26	3

6.2 How was CS integration with the curriculum?

In order to evaluate CS's integration with the curriculum we asked students to appreciate separately each of the project's components according to how useful and enjoyable each was. First, students were asked to classify between 1 (useless) and 4 (very useful) each of the components according to *how useful it was for their formation in this course*. Table 2 shows the results.

Table 2. Results of the multiple-choice question "How useful was each project component". Rating values are: 1 (useless); 2 (insufficiently useful); 3 (useful); 4 (very useful). Values represent the number of answers. The last column shows the number of blank answers.

	1	2	3	4	N/A
Reactive	3	5	41	28	2
BDI	0	0	26	52	1
Cooperation	1	6	31	39	2
Additional Topic	6	19	39	10	5
Tournament	17	30	23	4	5

As can be seen, all project's components, with the exception of the tournament, were perceived as useful for the students' formation.

6.3 How was the CS platform perceived?

In order to evaluate the platform itself, students were asked to classify it, between 1 (very poor) and 4 (very good), with respect to ease of learning, documentation quality, ease of programming and overall appreciation.

Table 3 shows the results. As can be seen, it is clear that the platform needs further improvements. In fact, even thought some examples of usage are provided, 50% of the students said that it was not easy to learn or to program. Also, the platform's documentation can be improved (89% of the students classified it as poor).

Table 3. Results of the platform rating question. Rating values are: 1 (very poor); 2 (poor); 3 (good); 4 (very good). Values represent the number of answers. The last column shows the number of blank answers.

	1	2	3	4	N/A
Ease of learning	19	30	24	4	2
Documentation	44	24	8	0	3
Ease of programming	9	35	25	8	2
Overall appreciation	12	38	25	2	2

7 Discussion and Conclusion

Results indicate that Counter-Strike is appropriate for our *Software Agents and Multi-agent Systems* course. In fact, most students seem to have enjoyed the project even though a significant percentage of which did not know the game before taking the course. These results are in line with our expectations that mainstream games engage and motivate students. Furthermore, most students thought the game was appropriate for this course suggesting that mainstream games can, in fact, be used as effective teaching tools.

Counter-Strike integration with the curriculum also seems to have been successful. Most students believed the project's objectives were useful for their formation and, at the same time, have enjoyed the project.

In contrast, the Counter-Strike platform still requires further improvement. Indeed, on a scale of 1 (very poor) to 4 (very good), about 65% of students graded the platform poorly. A clear weakness is the lack of proper documentation which should be immediately addressed.

Overall, the idea that mainstream games have a place in the multi-agents classroom seems to apply. Furthermore, our integration of this idea with the curriculum seems to have been successful. However, the realization of this integration through the Counter-Strike platform requires further work.

8 References

[1] Entertainment Software Association (ESA), "2006 Essential Facts about the Computer and Video Game Industry", www.theesa.com/facts, 2006.

[2] T. Malone, "Toward a theory of intrinsically motivating instruction.", Cognitive Science, vol.5, no.4, pp.333-369, 1981.

[3] A. Mitchell, C. Savill-Smith, "The use of computer and video games for learning – a review of the literature", Learning and Skills Development Agency, 2004.

[4] J. Kirriemuir, A. McFarlane, "Report 8: Literature Review in Games and Learning", Bristol: Nesta Futurelab, 2004.

[5] S. Tisue, U. Wilensky, "NetLogo: Design and Implementation of a Multi-Agent Modeling Environment", Agent2004 Conference, Chicago, IL, 2004.

[6] M. Fasli, M. Michalakopoulos, "Teaching e-markets through simulation games" in AAMAS 2005 Teaching MAS Workshop, 2005.

[7] TAC, "Trading Agent Competition", http://www.sics.se/tac, 2006.

[8] F. Bellifemine, A. Poggi, G. Rimassi, "JADE: A White Paper", TILAB "Exp in search of innovation" Journal, vol.3, no.3, pp.6-18, 2003.

[9] R. Adobbati, A. Marshall, A. Scholer, S. Tejada, A. Kaminka, S. Schaffer, C. Sollitto, "Gamebots: A 3D Virtual World Test-Bed for Multi-Agent Research", Proc. of the 2nd Intl. Workshop on Infrastructure for Agents, MAS and Scalable MAS, 2001.

[10] G. Kaminka , M. Veloso, S. Schaffer, C. Sollitto, R. Adobbati, A. Marshal, A. Scholer, S. Tejada, "GameBots: the ever-challenging multi-agent research test-bed", Communications of the ACM, Jan., 45(1), pp.43-45, 2002.

[11] Counter-Strike, "Counter-Strike Official Web Site", www.counter-strike.net, 2006.

[12] CPL, "Cyberathlete Professional League Homepage", www.thecpl.com/league/, 2006.

[13] CSports.Net, "CSports.net – Worldwide Computer Games Rankings, Top Games", www.csports.net/TopGames.aspx, 2006.

[14] Valve Developer Community, "Valve Developer Community Portal", developer.valvesoftware.com/wiki, 2006.

[15] HPB bot, "HPB Bot Homepage", botman.planethalflife.gamespy.com/, 2006.

[16] YapB bot, "YapB Bot Homepage", yapb.bots-united.com/, 2006.

[17] M. Wooldridge, "An Introduction to Multi-Agent Systems", John Wiley and Sons, 2002.

[18] M. Georgeff, B. Pell, M. Pollack, M. Tambe, M. Wooldridge, "The Belief-Desire-Intention Model of Agency", Proc. of the 5th Intl. Workshop on Intelligent Agents V: Agent Theories, Architectures, and Languages (ATAL-98), LNAI Volume 1555, pp.1-10, Berlin: Springer, 1999.