ABSTRACT
The importance of the video game and interactive entertainment industries has continuously increased in recent years. Academia has introduced computer game development in its curricula. In the case of the University of Cádiz, a non-compulsory course about “Video Game Design” is offered to Computer Science students. In the artificial intelligence lesson, students play a competition to earn part of their grade. Each student individually implements a strategy to play a predefined board-game. The different strategies compete in a software environment that implements it. Grading this part of the course depends on the results obtained in the competition. Additionally, before running the competition, students have to read the source code written by their classmates and bet part of their grade on other teams’ results. This way, they elaborate a critical analysis of the source code written by other programmers: a very valuable skill in the professional world. Students that are not so proficient in coding are still rewarded if they demonstrate they can do a good analysis of the source code written by others.

Categories and Subject Descriptors
1.2.1 [Artificial Intelligence]: Applications and Expert Systems; K.3.2 [Computers and education]: Computer and Information Science Education

General Terms
Experimentation

Keywords
Competition, video game development, artificial intelligence, expert system programming

1. INTRODUCTION
The importance of the video game and interactive entertainment industries has continuously increased in recent years [16]. Academia has introduced computer game development in its curricula. In the case of the University of Cádiz, a non-compulsory course about “Video Game Design” is offered to students of its degrees on Computer Science in their last year (third one). The recommended prerequisite is passing the Object Oriented Programming, Data Structures and Analysis and Design of Algorithms courses of the second year. But it is not a prerequisite as our experience gives all the necessary background on the topic and the course does not include lessons on expert systems.

The course mainly follows a project-based learning approach [13]. But in the artificial intelligence lesson, students play a competition (implementing a serious game [3]) to get part of their grade. Each student individually programs a strategy to play a predefined board-game. The different strategies compete in a software environment that implements it. Grading this part of the course depends on the results obtained in the competition, serving as a motivating factor [2].

To implement the competition, a specially designed free software tool provides a common environment where different strategies coded using rule-based expert systems compete against each other. The system models a square board in which players have a partial knowledge of the world (i.e., there is no “perfect strategy” to play the board-game). Using this application, the students can program and test their expert system rule sets against other rule sets or human players. After competing in a league, students can improve their systems before participating in a play-off tournament. This way, the students can develop the skill of critically analyzing the strategy they have implemented.
Additionally, besides programming a strategy, students have to read source code written by their classmates and bet a part of their grades on other teams. This way, they review source code written by other programmers, which is a very valuable skill in the professional world. At the same time, students that are not so proficient in coding are still rewarded if they can do a good analysis of the source code written by others by identifying the future winner of the competition.

The rest of the paper is organized as follows: first, we describe the methodology followed in the classroom. Then, we discuss the results obtained last year. The fourth section comments similar experiences. Finally, we compile some conclusions and draw future works.

2. METHODOLOGY

The main aim of the experience is that students of the video game design course learn artificial intelligence programming using expert systems. All work for this lesson must be done within the allotted 8 hours in the classroom, so the game has to be very easy to play and code for.

2.1 The game

The game implemented for the competition is a simplified version of the Stratego board game [19]. It pits two armies made up by sixteen pieces against each other in an 8x8 grid. Both armies have the same pieces listed in Table 1. Every piece has an associated value that is initially only visible to its owner. Players can initially place their pieces on the board in any order: one army is placed on the two lowest rows of the board and the other on the two highest rows. Figure 1 shows a possible initial set: brackets indicate that the value of the piece is hidden to the opponent.

Players take turns moving one piece to an adjacent square (upwards, downwards, rightwards or leftwards). When a piece moves into a cell occupied by an enemy piece, their value is revealed and the lowest valued piece is removed from the board (dies). If they have the same value, both pieces die. The objective of the game is to capture the opponent’s lowest valued piece. After a certain number of turns have passed, the match is tied.

A free software tool named Gades Siege [8] has been developed to support the competition. It improves upon a previous system called “Resistencia en Cádiz”, and provides a common environment where the expert systems developed by the students and those saved from previous years can face off each other. Gades Siege allows for playing leagues, playoffs and single matches step by step, or collecting statistics from batches of 100 unattended matches against an opponent (using different random seeds). A human can also take control of a team to test the behavior of the strategy under certain situations. This helps students test and improve their strategies.

Figure 2 shows a screenshot of a match between two teams. Most of the window is dedicated to the game board. The right edge of the window shows the names of both teams and their dead pieces, along with the number of turns played and the remaining turns before the match is tied. Below the game board, four buttons with arrows allow for navigating the saved game states for the match. From left to right, they allow for replaying the match from the beginning, go back one step, go forwards one step, and fast forward to the last turn, respectively. Finally, three more icons on the bottom-right corner activate automatic simulation (advancing one turn per second), continue to the next match (or exiting if not inside a tournament), and exit the tournament, respectively.

On the one hand, the game implements a partial knowledge world for players. Thus, there is no “perfect strategy” to play the game, avoiding a loss of interest due to unbeatable teams. But, on the other hand, players have perfect knowledge of their armies. Since there is only one agent playing, the game is kept simple as there is no need to implement the fuzzy intelligence and communication protocols usually required by a multi-agent system. As a result, the problem is suitable to be solved using expert systems.

In particular, as the board is discrete, the artificial intelligence can be easily programmed using a rule-based expert system [5]. This way, the strategy is implemented as a set of prioritized if-then rules that are executed according to information from the current game state. Programming is as simple as adding, modifying, removing or re-prioritizing rules. This offers a wide variety of behaviors that can be implemented making slight changes in a rule set. It includes non-deterministic behaviors if several rules of the same priority can be activated at a certain moment. Even more, while general rules seem good for the beginning of a match, more specific ones play better in the last steps or in certain situations. So a good balance between both kinds of rules is required to create a winning team.

The system is programmed in Python using PyGame (a wrapper of the SDL multimedia library), and Glade for the GUI. As for the core, it is developed on CLIPS [15]. All sys-

<table>
<thead>
<tr>
<th>Value</th>
<th>Number of pieces</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
tems and our own code are multi-platform and free. Gades Siege is freely available for download at [8].

2.2 Schedule

The experience was performed throughout 2 weeks, using the weekly 4-hour blocks available for the course (2 hours of lectures followed by 2 hours of laboratory):

- A 2-hour lecture dedicates the first hour to provide the required conceptual foundations behind rule-based expert systems, emphasizing their practical applications in science and engineering. The second hour is used to show the game. Some sample rules are shown along, from simple cases to a reasonably complex system, so they can get familiar with the environment.

- After the lecture students move to the computer classroom. Due to the ease to program the system, they can code basic rule sets in 2 hours, learning expert system programming with the support of a teacher.

- Students have 5 days to code their strategies at home. Two of the features previously commented proved to be very useful for learning: playing a very large batch of matches automatically versus sample strategies to collect statistics on general performance, and playing against a human to test how the strategy behaves under certain circumstances and do fine tuning.

- One day before the competition starts, students have to upload their strategies into the Moodle virtual learning platform used in the course. When all the students have uploaded their strategies the teachers make them publicly available for the enrolled student, so they can read the code of colleagues. Note that only strategies and not initial formations are available for download. If they both could be downloaded, the students could actually play a competition and (except for light differences caused by random seeds) know the winner.

- Lecture time for the second week starts with students declaring which strategy or strategies will bet part of their marks. They can bet on one, two or three different strategies, and the sum of the grades bet must be between 10% and 50%. Then, a two-legged round robin league in which all students are confronted against each other is played. Some matches are watched by the entire class on a large wall screen so students can explain the behavior they implemented.

- Usually, after playing the league, students identify minor issues and weak points in their systems that could be easily fixed. Note that a change in the priority of a rule or the addition/deletion of one rule more is very easy to implement and can lead to a very different behavior. This improvement could include adding rules considered suitable from the strategies of a classmate. They have 1 hour in the laboratory to modify their rule set and then, the final strategies take part in a playoff tournament.

2.3 Grading

In order to pass the course, a strategy only needs to defeat a naive (sparring) team that is included in the system. This way, students can be confident they will pass this part of the course, even if the rule set ends up last in the competition. We do not consider fair that students able to program a good expert system in 2 weeks fail because their classmates are better programmers. Regardless of the quality of the strategies, one student must necessarily win the tournament, another must be second, and so on until the last place.

An additional point is given if the student explains clearly her strategy. This prevents cheating on one hand (or at least, it ensures that students know what their strategies do), and on the other hand, classmates can easily use some rules to improve their strategies for the play-off. In order to further improve their marks, students need to win matches against their classmates.
Performance of the team

<table>
<thead>
<tr>
<th>Mark (0-10)</th>
<th>Performance of the team</th>
<th>Range of teams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fail (below 5)</td>
<td>No match won</td>
<td>0 or 1</td>
</tr>
<tr>
<td>5</td>
<td>Only defeats the sparring (DS)</td>
<td>0 or 1</td>
</tr>
<tr>
<td>6</td>
<td>DS and explains the strategy clearly</td>
<td>0 to All</td>
</tr>
<tr>
<td>7</td>
<td>DS and wins another match</td>
<td>0 to 2</td>
</tr>
<tr>
<td>8</td>
<td>DS and stays in upper half of the league or passes one play-off round</td>
<td>Half to all</td>
</tr>
<tr>
<td>9</td>
<td>DS, stays in upper half of the league and passes one play-off round</td>
<td>Half to half+2</td>
</tr>
<tr>
<td>10</td>
<td>DS, stays in upper half of the league, passes one play-off and wins a tournament</td>
<td>0 to 2</td>
</tr>
</tbody>
</table>

3. RESULTS OF THE EXPERIENCE

Due to the nature of the game (providing only partial knowledge of the world) and the limited knowledge of the undergraduate students on artificial intelligence techniques, there has been no “absolute best” team in any year. It is curious that quite often some of the teams in the top of the league had difficulties or even could not beat some in the lowest part. This demonstrates that even the best teams had some weak points.

Informally speaking with students, they reported that the competition between peers is a motivating factor to learn expert system programming, reinforcing conclusions of similar experiences [15]. They also commented that seeing the army evolve without their intervention against others was a nice way to understand the application of artificial intelligence to game development.

In this year 15 students joined the course, but only 14 attended the experience: a student uploaded a strategy but did not go to the sessions and got 0 points in the bet (as he did not provide a satisfactory reason to miss it). The 14 students bet an average of 24.29% of their marks: most bet below 30% except for one who bet 40% and another one who bet 50%. As for the lowest bets, three students bet the minimum 10% of their grade.

Only three students obtained lower grades due to their bets and seven improved it. The maximum gain was obtained by a student whose strategy had 7 points and bet half of his mark for the winning one (who obtained 8, the maximum available in the league). Highest losses were −0.85 and −0.65. The rest of students varied their strategy grades in a 0.2 points or less. Finally, four kept the same grade because

Strategies get extra points (up to a total score of 10) for each of:

- Winning a match (not against the sparring opponent).
- Being in the top half of the league.
- Passing a play-off round.
- Winning a tournament.

The experience up to this point has been carried out since the first year of the course, in 2007–08. It has a clear problem: it implies certain restrictions on the number of teams that can get certain grades. For example, there can be only one winner in each competition, and only half of the teams will pass the first play-off round. This limits the distribution of grades according to teams’ performance as shown in table 2.

Table 2: Grades distribution

<table>
<thead>
<tr>
<th>Mark (0-10)</th>
<th>Performance of the team</th>
<th>Range of teams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fail (below 5)</td>
<td>No match won</td>
<td>0 or 1</td>
</tr>
<tr>
<td>5</td>
<td>Only defeats the sparring (DS)</td>
<td>0 or 1</td>
</tr>
<tr>
<td>6</td>
<td>DS and explains the strategy clearly</td>
<td>0 to All</td>
</tr>
<tr>
<td>7</td>
<td>DS and wins another match</td>
<td>0 to 2</td>
</tr>
<tr>
<td>8</td>
<td>DS and stays in upper half of the league or passes one play-off round</td>
<td>Half to all</td>
</tr>
<tr>
<td>9</td>
<td>DS, stays in upper half of the league and passes one play-off round</td>
<td>Half to half+2</td>
</tr>
<tr>
<td>10</td>
<td>DS, stays in upper half of the league, passes one play-off and wins a tournament</td>
<td>0 to 2</td>
</tr>
</tbody>
</table>

As remarked before, the formulas concern only the league, covering up to 9 points. Two additional points could be obtained in the play-off of the improved strategies, up to 10 total points.
they bet different rates for several teams whose weighted mean was the same grade their strategy got.

Interestingly, some students acknowledged that the main reason to bet for a certain team was not only their impressions when they reviewed the code, but the overall marks of the developers in previous courses. Although we do not have access to that information, in general seems that students who perform well in programming subjects, also do well in this experience.

Students were, on average, much more interested in this course than in others, judging from the anonymous surveys conducted near the end of the term. It included different aspects about the course, all of them rated from 1 (minimum) to 5 (maximum). Results showed a very high score:

- Evaluation method in the course: 4.69
- Competition of expert systems: 4.92
- Betting system included: 4.15
- General course value: 4.61

4. RELATED WORKS

There are similar experiences of competition between students in the literature. Concerning simulated robotic soccer, there are different experiences using RoboCup Soccerserver, an evolution of Javasoccer [10]. This systems uses a client-server architecture, so the artificial intelligence modules can be programmed not only using rule-based expert systems [12] [4], but also other techniques [11] [17]. In 7 groups of students had to implement the artificial intelligence for a RoboCup team during a course. At the end of it, a competition was run among the teams coded by students and some other from Robocup World Cup participants. The basis for the student’s examination and grade was a written report on the team design and performance in the competition. The

Another example is Robocode [6]. Robocode is a game that implements a virtual battlefield where two robots fight each other. Students use Java to implement different artificial intelligence techniques. Then, they observe how their agent performs compared to others by watching the agents compete.

There is an interesting experience using software based on the Ataxx game [14]. Groups of two or three students had to use Prolog to implement an agent which had to complete every game using a limited amount of CPU time. The assignment grade was determined by the ranking obtained in a free-for-all competition among these student programs.

Finally, even hardware competitions [9] [11] using robots are documented. They are developed in the laboratory sessions of the Artificial Intelligence course. This way, students can bring into practice the concepts learned and are highly motivated.

Nevertheless, all of the experiences commented use much more complex games than our proposal. So they would have needed more time than what was available. As we commented previously, time restrictions are very tight in this experience. As for the betting system, we have found no experiences using it, showing its originality.

5. DISCUSSION AND CONCLUSION

The importance of the video game and interactive entertainment industries has caused academia to introduce computer game development in their curricula. In an elective “Video Game Design” course offered in the Computer Science degrees of the University of Cádiz the grade in the artificial intelligence lesson is obtained competitively. Students code strategies to play a serious game. They compete in a software environment that implements a board-game, and the results obtained in the competition will be part of their grade.

Firstly, each student has to code a strategy as a rule-based expert system. Those strategies compete in a league. Then, students can improve them before participating in a play-off tournament. Another part of the grade depends on a bet on the strategies of their classmates. This way, they practice critical analysis on the source code written by other programmers, a very valuable skill in the professional world.

At the same time, students that are not so proficient in coding are still rewarded if they demonstrate they can do a good analysis of the source code written by others identifying the future winner of the competition.

The experience has been rewarding both for the students and the teacher. Developing the environment was a time-consuming task in the first year, but results showed it was a very interesting way to motivate students. As the system is open source, is has evolved year after year, mainly due to students who detected deficiencies or interesting features that could be added and coded them. Thanks to its automation, the experience is very easy to handle. Enrollment figures were 20 students in 2011, 16 in 2010, 27 in 2009 and 13 in 2008.

Skills developed include not only those specific for the lesson (programming rule-based expert systems, using it for video game artificial intelligence systems, etc.) but also general ones: improvement of a strategy by analyzing its performance and critical analysis of the source code written by other programmers.

In general, due to the generous grading system used for the strategies, all the students get good grades in this lesson. This way, students that are able to program a good expert system in 2 weeks cannot fail simply because their classmates are better. This is necessary, as regardless of the quality of the strategies, the range of teams in each place of the tournament is limited: one student must win the tournament, another must be second, and so till the last place.

The use of a betting system has an interesting impact in the experience, as it relaxes the relation between the performance of the strategy and the grade obtained by its author. In fact, most of the 15 students in the course increased their grades thanks to it (seven in particular), and only three decreased them. Students could bet from 10% to 50%, being the average almost 25%. Interestingly, some students acknowledged that the main reason to bet on a certain team was not only their impressions when they reviewed the code, but the overall marks of the developer in previous courses. Although we do not have access to that information, our general impression is that students who perform well in programming courses also do well in this experience.

Two lines of future work lie ahead. The first line of work is making it easier to change the rules of the board-game for each competition or match without having to code them, so the teams could play in different environments and avoid
over-specialization. For example, there might be some unreachable square in the matrix that needed extra intelligence to play, or some pieces could be uncovered from the beginning of the match. Our second line of work would be applying log analysis techniques to detect weak rules in an strategy.

6. ACKNOWLEDGMENTS

This work has been funded by the Proyecto de Innovación Educativa Universitaria del Personal Docente e Investigador “La competitividad y el análisis crítico en la evaluación” (CIE44) of the Convocatoria 2010 de Proyectos de Innovación Educativa Universitaria for the PDI of the Universidad de Cádiz, funded by the Consejería de Innovación, Ciencia y Empresa of the Junta de Andalucía and the University of Cádiz (Spain), and by the Computer Languages and Systems Department of the same University.

7. REFERENCES