Fun in CS2*

Amalia Duch, Jordi Petit, Enric Rodríguez-Carbonell and Salvador Roura
Departament de Llenguatges i Sistemes Informàtics,
Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain

Keywords: Data Structures, Algorithms, Computer Game, Computer Game's Player, Programming.

Abstract: We report our experience of including the implementation of a player of a computer game as a programming project in a CS2 course focusing in data structures and algorithms. Every semester, an instructor designs the rules of a game, prepares its visualization, and implements an elementary player with a very simple strategy. The game is then delivered to students who, as a first step in order to pass the project, must program a player that wins the elementary player. Then, a tournament begins among all the students with accepted players. At every round of this tournament a player is eliminated until just one player, the champion, survives. Grades for this assignment are computed automatically and increasingly with respect to the round where students have been eliminated. The result is a fun and very motivating programming experience for our students.

1 INTRODUCTION

As professors of algorithmic subjects, we face nowadays that, frequently, our students are poorly motivated by our class matters but pretty hooked to computer games. So, following the saying *"If the mountain will not come to Muhammad, then Muhammad must go to the mountain"*, we have decided to introduce computer games in our data structures and algorithms courses.

We have done so by means of a programming activity consisting in the implementation of a strategy for a player of a computer game. The goal is that the programming of this player should involve the implementation of some of the algorithms and data structures taught during the course, and to motivate for the self study of others.

Here, to program a player consists in designing strategies to move some tokens, having complete information of the map in which the tokens are lying and about the rest of the players. In fact, each player controls a set of configurations that will interact with the game by reading information about the game world and inform of its moves to interact with it at each turn.

Note that games are meant to be played by programs, and to be seen by humans. There is no human interaction during the games. The program to control the intelligence of a player is written before each match starts.

The program submitted by a student (a player) will be faced to the programs submitted by other students, and will get better or worse score depending on the way it is planned, i.e., how good is in practice its programmed strategy. Unlike most exercises of the course, this project is open: it is not about solving a specific problem, but to prepare a strategy, to study the strategies of the opponent players, and, if required, adjust or reprogram the own strategy.

The idea was inspired by the ICPC-Challenges organized by ACM during the World Finals of their annual ACM International Collegiate Programming Contest[2]. A similar initiative is the AI Challenge[3].

The performance of this lab assignment encourages, among other skills, algorithmic programming aspects (implementing their own strategy), programming related to artificial intelligence (controlling the heuristics of the agents that appear in the game) and even good programming habits, because the players are expected to change over the project duration to adapt to their peers' strategies, and thus, the students with the most flexible and maintainable code will be benefited.

There are several other initiatives that include automatic evaluation and assessment systems, digital and web support, competitive and problem oriented

*Work supported by the Generalitat de Catalunya's project ALBCOM (ref. 2009 SGR 1137).

[2]See http://icpc.baylor.edu.

[3]See http://aichallenge.org.

Duch A., Petit J., Rodríguez-Carbonell E. and Roura S..
Fun in CS2.
DOI: 10.5220/0004389604370442
In *Proceedings of the 5th International Conference on Computer Supported Education* (CSEDU-2013), pages 437-442
ISBN: 978-989-8565-53-2

learning, among others (Giménez et al., 2012; Douce et al., 2005; Saikkonen et al., 2001; Joy et al., 2005; Cheang et al., 2003; Kosowski et al., 2007; Kurnia et al., 2001; Revilla et al., 2008). But, as we will show, thanks to the enthusiasm that video games rouse amongst students, the experience presented in this paper is particularly engaging.

2 CONTEXT AND GOALS

This project is included as part of the "Data Structures and Algorithms" course of the Facultat d'Informàtica de Barcelona and the "Algorithmics" course at the Facultat de Matemàtiques i Estadística (both at the Universitat Politècnica de Catalunya). These courses (with minor differences between them) start by introducing the concepts of algorithm analysis, together with the required basic mathematical tools. Then, these bases are used to study and analyze various implementations of classic and essential algorithms and data structures (sorting and searching, divide and conquer, dynamic programming, hashing, balanced binary trees, graph traversal, shortest paths, among others).

These courses intend to combine basic theoretical aspects of algorithm design and analysis together with several programming features. They consist of 6 ECTS[4] that involve two hours of theory classes, one hour of classes of problems and one hour of laboratory classes per week.

Inside such courses, the educational objectives of the game are the following:

- To provide an original and motivating environment to facilitate the integration of generic and technical skills.
- To supply an active learning activity that improves the learning of programming, data structures and algorithms.
- To encourage the development of competencies such as the design and analysis of effective, efficient, collaborative and/or competitive strategies to obtain well-defined objectives.
- To introduce didactic materials in new formats using new technologies (web, social networks, forum, graphics, etc.).
- To promote the programming and the design of efficient algorithms, improving the integration of practical and theoretical learning.
- To offer tools that allow students to strengthen competencies such as the gradual and continuous improvement and/or overcoming of previous work at both personal and social levels.
- To promote new evaluation mechanisms for the skills of programming and design of efficient algorithms.

[4]ECTS stands for European Credit Transfer System, which is a standard unit for comparing the study attainment of students of higher education across the European Union. One academic year corresponds to 60 ECTS-credits that are equivalent to 1500–1800 hours of study, i.e., 25–30 hours of study per credit.

3 DEVELOPMENT

The development of this activity has five main phases: preparation, distribution, qualification, tournament, and grading.

Preparation. A new game is created by one or more instructors of the course; some actual games are presented later, in Section 4. A game consists of the following elements:

- The *documentation* gives a basic overview of the game, describes its goals and rules, and presents a tutorial on how to use it.
- The *interface* of the game is the C++ API that users will use to write their own strategies. It basically includes functions to query the current state of the board and functions to order changes to the agents in the board.
- The *implementation* of the game includes all necessary coding to make the game work, such as reading the initial boards, handling the rules, updating the score, etc. It also includes the *viewer* of the game, that is, the tool that will be used to see games in a graphical way. For portability reasons, this viewer is programmed in Javascript.
- An elementary player called *the dummy*, which implements a fairly simple game strategy.

Distribution. All the above elements (including the source files) are given to the students, with the exception of the dummy player, for which only the object files are distributed. With this material, all students can design, implement, run, debug and view their own games at home as often as needed. In addition, students are given access to an online web server that will handle their submissions and games.

Moreover, we try to motivate students as much as possible by designing advertisement posters of the game. We hang them on the walls of classrooms and campus squares.

Qualification. After distribution, students are given three weeks to work on their own designing a strategy and writing a program for their players. The only requirement for their players to qualify is to consistently beat the dummy player. This requirement is necessary to ensure that a minimal coding effort is done by each student; otherwise even an empty strategy would qualify. In order to prove that they beat the dummy player, students submit their programs to our online judge, which checks that in four random games against dummy players, the submitted player always wins. Students have an unlimited number of opportunities to beat the dummy player within the three weeks deadline.

Tournament and Grand Final. The most exciting phase of the game consists in the play-off tournament, which starts after the previous deadline with all the qualified players. Over a couple of weeks, the game's website automatically takes care of the tournament, handling matches among subgroups of students (typically four) and performing the necessary rounds to give a ranking to all the players by successively eliminating the "worst player" of each round. All the games and results disputed during the tournament are publicly visible in the game website.

From round to round, students may still submit new strategies that will replace their formers ones, in order to try to react to the actions of their mates. These replacement players must, of course, also beat the dummy.

When only a few (16, say) players stand, a grand final is organized in the conference room of the Facultat d'Informàtica de Barcelona to discover the final champion. The surviving players are grouped into semi-finals, and the winners of each semi-final dispute the last round, where only one of them survives. During the ceremony, the programmers of these best players are invited to give a public short speech explaining how their players work and how they are programmed.

Grading. The total grade for our courses is a number between 0 (nothing) and 10 (perfect). Out of these 10 points, 9 come from standard evaluation systems (tests, exams, lab assignments). Then, all the students who beat the dummy before the deadline obtain one point. In addition, and according to their final ranking in the tournament, each student receives up to an extra point to be added to his final grade (for a maximum of 10, of course). This is computed proportionally to the time their player is still in the game, thus the first eliminated player gets no extra point while the champion gets one extra point.

4 ACTUAL GAMES

Over the past few semesters, we have created several games, trying our best to get attractive and enjoyable activities. All the games feature a board where several agents controlled by four different players interact for several rounds to get a final score. See Figure 1 for some examples.

PacMan. This is an adaptation of the classical arcade game for four players, where each player controls his own pacman and three or four ghosts. Normal pills eaten by pacmen score points, and power pills temporally boost pacmen to move faster and to eat ghosts, in order to get extra points and weaken adversaries.

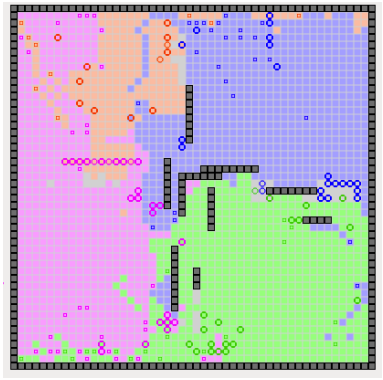
Battle Royale. Each player controls several knights and peasants. As they move to adjacent cells, peasants colonize cells. Knights capture (kill) adversary peasants to convert them to their team and probabilistically fight adversary knights. For each player, the score is the final number of colonized cells. The name of the game comes from the famous Japanese cult film, just for the large number of kills during each game.

Apocalypse Now. In this adaptation of the Vietnam war film, the goal of each player is to control as many check points as possible. To do so, each player can use soldiers, helicopters and parachutists, who have different mobility attributes depending on the terrain they move (jungle, field, water or mountain) and have different kinds of attack (body to body or napalm).

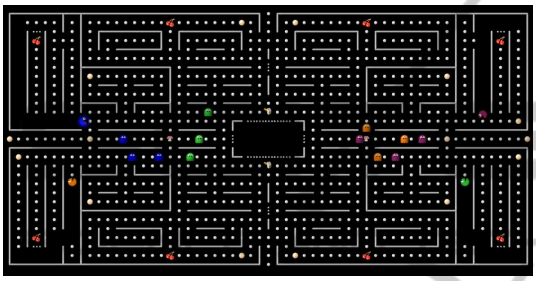
Tron. This is a variation of the classic game and film where each player controls four light cycles that should not crash with the prefixed walls in the maze nor with trails of light that all light cycles leave behind. In case of a crash, the light cycle and its trail disappear from the board. The winner is the last player that survives.

Dragon Ball. Inspired by the famous animated series, in this game each player must control Son Goku in his quest to collect dragon balls and bring them to hoi-poi capsules. In the process, players can launch kame hame attacks to other players and use kinton clouds to move faster.

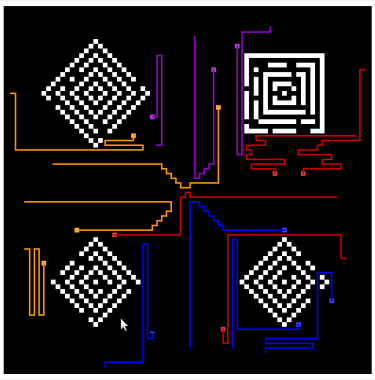
In all the games, the board is organized as a collection of cells that induce a graph. At each round, the movements of the agents are governed by each player strategy and invoked through the game API. Each player decides its movements for the next round independently of the other players, and there is a randomization process by which possible collisions are resolved.



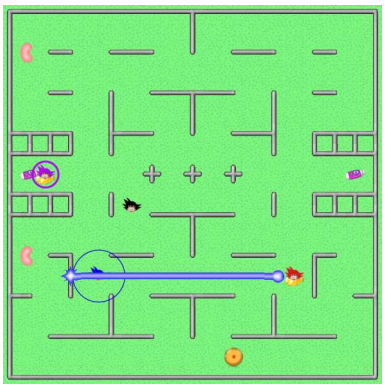
Battle Royale



PacMan



Tron



Dragon Ball

Figure 1: View of some games.

5 THE ONLINE SYSTEM

Students use a website to submit their solutions, check whether their programs beat or not the dummy, request training games with other players (so that they do not need to exchange their source code to play games among them) and track the results of the tournament games, which presumably determines the quality of their programs. This website is largely independent of the game, so is reused each semester.

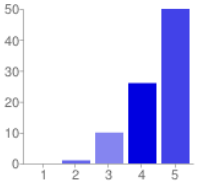
To dispute the games, our system uses the facilities of the Jutge.org infrastructure, which is an educative online programming judge developed at our University and open to all users (Giménez et al., 2012). In brief, Jutge.org offers a highly scalable and secure system that processes submissions to programming assignments and offers a verdict on their behavior.

The techniques to dispute our types of games must however extend the security measures used in similar online judges (Forišek, 2006), because not only they must prevent players to abuse or tamper the system, but they also must prevent to cheat other players. To do so, our system disputes the games by confining each player to its own process with its own user. Moreover, there is an additional process that mediates in their communication (to check that they obey the rules of the game) and supervises the player processes (to check that none exceeds memory or time limits and to handle this case if needed).

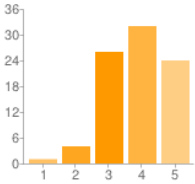
While these levels of security can seem paranoid at first, they are rather necessary, as some students are able to devise contrived ways to try to cheat. For instance, during the last semester, a group of students has used knowledge on the way that the parameters are pushed on the stack of function calls to locate the memory positions that encode the walls of the maze, and so could change them to their advantage. While this hack worked with the distribution of code we give to students and could amaze (or even terrify) some of their opponents, it did not have any consequences inside the server that plays the official tournament thanks to not sharing the memory space.

In addition, our system interacts with the well-known JPlag service to detect (and specially discourage) plagiarism (Prechelt et al., 2007). This could be an important issue (it is not), because none of the submitted programs will normally be read.

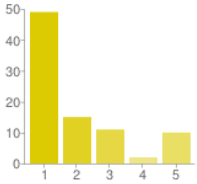
As an example of the magnitude of the tournaments, in the semester PacMan was played, we had 143 students that submitted 2929 programs, 749 of them beating the dummy player. A total number of 7761 games were disputed and 2.2 GB of data storage was needed.



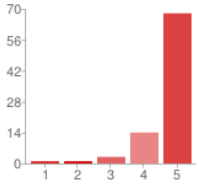
I liked the game.



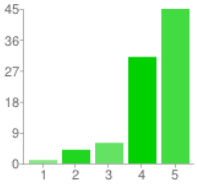
It was easy to beat the dummy player.



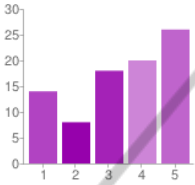
I would rather spend my time studying theory than programming my player.



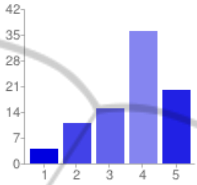
To program a player is much more motivating to me than traditional programming projects.



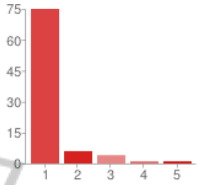
Competing against my classmates in the game motivates me.



I have taken the time to improve my player after winning the dummy player.



The documentation of the game is right to me.



I would rather prefer this course without game.

Figure 2: Results of the survey for the PacMan game (1: Strongly disagree → 5: Strongly agree). Values are in percentages.

6 SUPPORT

It may be clear now that, besides the investment of building an online system to process submissions, the development of this activity repeatedly calls for some effort of the course instructors.

This support starts at the preparation phase (we estimate that this involves about 40 hours of work), continues monitoring the game and reacting in the case of need (we estimate this involves about 10 hours of work) and finishes preparing the grand final.

Indeed, as most participants and many of their classmates look forward to the grand final of the game, this is an important event that is well prepared in advance: slides, game matches, videos, viewers, etc. The final is attended by many students and faculty staff. It is worth commenting also that in order to increase the students excitement and expectation about the grand final, we design a new visual interface (as attractive as possible) together with a stylish soundtrack. A video recording the grand final of the Battle Royale game can be found at http://media.fib.upc.edu/fibtv/streamingmedia/view/2/209. At the moment, this is the video with most views among all videos posted in our computer science school.

In addition, one has also to add up the time to give support to the students that request help from their instructors. Some of them require individual or group office hours.

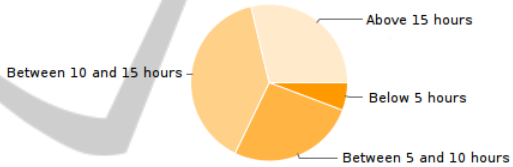


Figure 3: Results for the question *Time I have spent working on my player*.

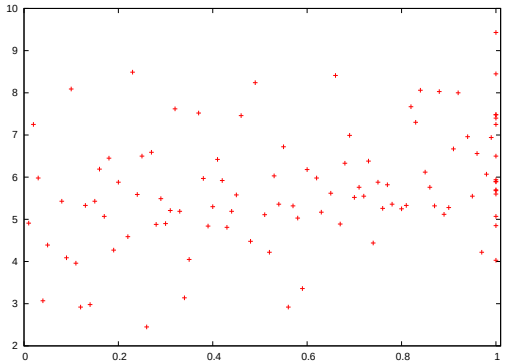


Figure 4: Scatter plot of final grade (from 2 to 10) according to rank in the PacMan tournament (from 0 [worst] to 1 [best]).

7 RESULTS

The game activity has been implemented as described in the latest five semesters. It got a large percentage of participation (always more than 90%) and almost all

students who tried it could enter a player into the tournament, i.e., their player could win the dummy and therefore, they passed the project. Moreover, since to keep alive during the tournament provides extra points for the course's grade, this activity really helps several students to pass the course (about a 20% of students pass the whole course because of this extra point).

In order to support these statements, last semester we applied a survey to our students. Figures 2 and 3 show the results of each of the issues on which we consulted students. The range of answers was from 1 (strongly disagree) to 5 (strongly agree). From the surveys, we can infer that almost all students enjoy the game activity, that they do not find too many problems in beating the dummy player, and that they prefer it to more standard assignments. Also, they like to compete against each other. Therefore, we have corroborated that presenting the project as a game has had a great impact on teaching. Moreover, this practice model is very attractive and motivating to our students.

On the other hand, one could think about some correlation between the ranking of the students in the tournament and their final grades in the course. Figure 4 shows this information, and the Spearman's rank correlation concludes that although the correlation is weak (Rho $= 0.38$), the correlation is statistically significant (p-value $= 2.757e-05$).

As said, during the grand final, the programmers of the best players explain their strategies. According to their explanations, in a few cases it is possible to reach the grand final with just a very simple player, but in general, winners implement smart and complicated strategies, with sophisticated algorithms.

8 DISCUSSION

Along this work we have described our experience introducing programming computer strategies for computer games into typical CS2 courses. We have shown evidence on how this kind of programming activity is fun and highly motivating to computer science and mathematics students. This motivation also encourages professors, and thus facilitates a pleasant working environment.

As a weak point of such a project, one can think that students seem to be so hooked that they spend more than the recommended hours to program and improve their players, and this in detriment of the hours they should dedicate to other parts of the course and to other courses. But, when surveyed about this issue, they did claim that they dedicated a big amount of time and work, but that this was mostly in their free personal time, which otherwise they would not have dedicated to study.

Overall, we consider it such a successful experience that we want to continue improving and spreading. Among the ideas we have to go further, it is worth mentioning that the games we have created so far are very competitive. So, in the future we would also like to develop collaborative games in which players should help each other in some way. On the other hand, we want to extend our online system to offer these games to the general public. A systematization of the website and of the design process of new games is under current development.

ACKNOWLEDGEMENTS

We thank Omer Giménez and Mario G. Munzón for all their enthusiasm, ideas and programming time.

REFERENCES

Cheang, B., Kurnia, A., Lim, A., and Oon, W.-C. (2003). On automated grading of programming assignments in an academic institution. *Computers & Education*, 41(2):121–131.

Douce, C., Livingstone, D., and Orwell, J. (2005). Automatic test-based assessment of programming: A review. *ACM Journal on Educational Resources in Computing*, 5(3).

Forišek, M. (2006). Security of Programming Contest Systems. In Dagiene, V. and Mittermeir, R., editors, *Information Technologies at School*, pages 553–563.

Giménez, O., Petit, J., and Roura, S. (2012). Jutge.org: An educational programming judge. In *Proc. of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE-2012)*, pages 445–450. Association for Computing Machinery.

Joy, M., Griffiths, N., and Boyatt, R. (2005). The BOSS online submission and assessment system. *ACM Journal on Educational Resources in Computing*, 5(3).

Kosowski, A., Malafiejski, M., and Noinski, T. (2007). Application of an online judge & contester system in academic tuition. In *ICWL'07*, pages 343–354.

Kurnia, A., Lim, A., and Cheang, B. (2001). Online judge. *Computers & Education*, pages 299–315.

Prechelt, L., Malpohl, G., and Philippsen, M. (2007). JPlag: finding plagiarisms among a set of programs.

Revilla, M., Manzoor, S., and Liu, R. (2008). Competitive learning in informatics: The UVa online judge experience. *Olympiads in Informatics*, 2:131–148.

Saikkonen, R., Malmi, L., and Korhonen, A. (2001). Fully automatic assessment of programming exercises. *ACM SIGCSE Bulletin*, 33(3):133–136.