Serious Games: Using Abstract Strategy Games in Computer Science 2

An experience report and lessons learned

Inés Friss de Kereki, Alejandro Adorjan
Facultad de Ingeniería
Universidad ORT Uruguay
Montevideo, Uruguay
kereki_i@ort.edu.uy, adorjan@ort.edu.uy

Abstract—In this paper, we describe the use and lessons learned of "serious games", in particular "abstract strategy games", at freshmen-level in Computer Science 2 (CS2) course. In this course, the goals are that the student can analyze non trivial domains using methodology; design solutions, and implement them under the object oriented paradigm (OOP). Since some years ago, we included this kind of games as a tool to teach the main concepts of the course. Here, we describe the CS2 course and abstract strategy games, including how to use them in the course and some examples. Moreover, we collected students' results and opinions, and also teachers' perspective. According to these data, its use is highly motivating and promotes learning. We also offer some lessons learned.

Keywords—Computer Science 2, Programming, Games, Abstract Games

I. INTRODUCTION

Different pedagogical approaches are being used by the academy in order to motivate, recruit and retain students to provide them high quality education that prepares them as professionals. Engaging future engineers is a central topic in Engineering Education, where teachers must adopt different strategies and methodologies in order to understand the complex educational context [1]. Several teaching-learning approaches are emerging, among others: Flipped-Classroom, Peer-Instruction, Just In Time teaching, Blended Learning, and the use of Educational Games.

"Engagement in education remains a challenge that can potentially be addressed through Game Based motivation in learning interaction" [2]. Educational games can provide students with a motivating and stimulating environment, as refer Bodnar et al [3]. An educational game has to succeed on two fronts: as an educational tool and also as a fun game [4].

This paper presents the use and lessons learned of "serious games", in particular "abstract strategy games" (ASG), at freshmen-level in Computer Science 2 (CS2) course. In this course, the goals are that the student can analyze non trivial domains using methodology; design solutions, and implement them under the object oriented paradigm (OOP). Since some years ago, we decided to include this kind of games as a tool to teach the main concepts of the course.

In detail, we describe the characteristics of the CS2 course and ASG, including how to use them in the course and some examples. We collected students' results and opinions, and also teachers' perspective. So, the paper is organized as follows: in Section II we describe serious games. In Section III, we focus on our Computer Science 2 course. In Section IV, we include the use of games in CS2. In Section V we present students' results and surveys. Finally, we present lessons learned and conclusions in Section VI.

II. SERIOUS GAMES

Serious games are computer applications, which combine the serious aspects with the playing aspects [5]. Those games have great potential, they can take benefit of the intrinsic motivation of people to play and have fun, but is not obvious that if they effectively achieve their purpose [6]. There are an increasing number of research studies in Computer Science education related with serious games. Several works report that its use is motivating and enhance learning [7][8][9][10].

An "abstract game" is a game with no theme or in which the theme is not relevant. Also, abstract strategy games minimize the randomness. There are no non-deterministic elements (such a dice roll or cards). There are no hidden information: "each player, when deciding his move, must have complete information about the current position of the board" [11]. As refer Heliotis et al, "board games are typically discrete, where the game state can be stored in basic data structures" [12]. After a finite number of alternating turns, the game ends.

Some examples of ASG are: Kulami [13], Morelli [14], Pentago [15], FlipFlop [16] and Quinamid [17].

In detail, Kulami [13] includes two players that place their marbles on a game field of interconnecting wooden panels. The game includes 17 wooden panels of various sizes, for example 4 x 6 fields, and 5 x 4 fields; 28 red and 28 black glass marbles. The panels can be arranged in different order, so the game field may be not regular (see Fig. 1). Each player strategically positions his or her colored marbles (red or black) in the free space in a panel. The rules are: "row/column rule": the last marble set in a panel determines the row or the column of the next marble of the other player, and "No Use rule": the last 2 panels used can not be used for the next play. The panels used before can be reused.A player wins the "control" of a panel if he or she has the majority of marbles of that panel of his or her color. Points are obtained by controlling panels and also by forming lines of the same color. The goal is to obtain the maximum points. The game ends when all the marbles were put or it is not possible to put more according to the rules.

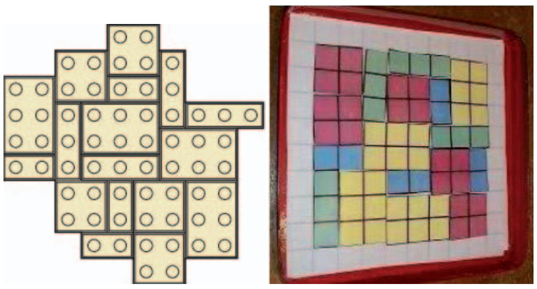


Fig. 1. Kulami: irregular [13] and regular board game

In Morelli [14], there are two players with 24 reversible black/white stones and a tower each. Each player has one color. It is played on a square board with concentric squares, each one of a different color. The stones are disposed initially on the red band. Players take turns to move one stone of their color at a time. See Fig. 2.

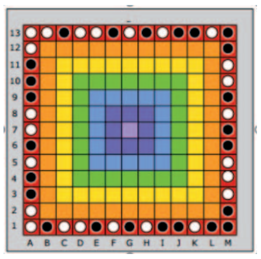


Fig. 2. Morelli: board [14]

The winner is who occupies the center with his tower at the time of finishing the game. If at the end of the game the center was not occupied, it is a draw. The game ends when a player has no possible movement or when a number of pre-established moves are made.

A move consists of three parts: first moving, then capturing, and finally verifying the center. The player move a piece of its own over one or several empty squares diagonally or orthogonally, to an empty position in a concentric square closer to the center than the concentric square of departure. It is not valid to put a stone in the center. At the end of that movement, captures are automatically made if there are.

A capture is given when at the end of moving the stone, from the final position, consecutive neighboring stones of the other color are "locked" in line and at the other end there is one of their own, in a similar way as the well known game “Othello” or “Reversi”. In that case, the intermediate stones are turned over, so they get the player's color. There may be multiple captures with the same stone. There are no captures “in chain”. The tower does not participate in captures and can not be moved.

Later it is verified automatically if the center is occupied. If a “frame” is formed, the center is occupied. A frame consists of 4 player stones, which form a perfect square whose center is the center of the board itself. In that case, the player's color tower is placed in the center. If there is already a tower in the center of the other color, it is changed to the corresponding color. Frames can not have ends on the outer edge of the board being played.

Pentago [15] is an abstract strategy game for two players with four 3×3 grids arranged into a larger 6×6 grid. The objective of the game is to be the first in obtaining five own marbles in a row. Each player, in his/her turn, places a marble and has to twist one of the grids by 90°. See Fig. 3. If lines of both colors occur simultaneously, the player who made the last move wins. If the board was completed and there is no case of 5 in line of the same color it is a draw. To add complexity, the game could be played in 2 versions: with compulsory rotation or not.

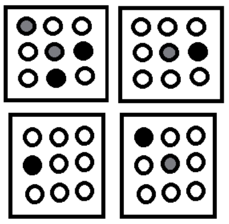


Fig. 3. Pentago: game in course

In Flip-Flop [16], first the board is chosen (grid of 3*3 or 5*5). Each of both players has 3 cards or pieces (for the case of 3*3) or 5 pieces (for the 5*5 board). The pieces are of their color and have on one side the letter "T": tower and in the other: "B": bishop.

As in chess games, the tower moves in a straight line, horizontally or vertically and the bishop moves in a straight diagonal line. The white player puts his cards on the lower edge with the "T" visible. The black player puts his cards on the top edge with the "T" visible. In the center of the respective edge, there is the goal of each player (see Fig. 4). A player wins when at the end of the opponent's turn, he has some of his own pieces in the other player's goal, or when the opponent has no valid movements. The white player starts the game and they alternate the turns. On his/her turn, each player moves one of his pieces following these rules:

- You can move any distance over empty places.
- You can not skip other pieces.
- When the movement is finished, the card is inverted (if the “T” was shown, then the “B” is shown, and vice versa).
- You can not pass the turn.
- You can capture an opponent's piece only if that piece is in a goal (your own or someone else's). To capture it, you must move your own piece (according to the rules explained) to that goal and the piece of the other player is removed.
- Pieces are not re-entered in the board.
- If there is a piece of the opponent in the own goal, it is mandatory to try to defend the goal. If you can not defend, you lose.

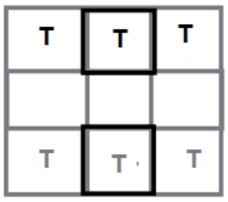


Fig. 4. Flip-Flop: initial setup for 3*3 game

Quinamid [17] is an abstract 2-player game in which the objective is to create a line of 5 counters. The twist here is that the counters are placed in a series of 5 boards of differing size that are formed to make a pyramid." (see Fig. 5). As an extension, at the beginning of the game, the length of the line to be formed and the dimension n of the largest board could be selected.

In each turn, the player can: a) place a counter in a vacant space, b) rotate a board (including those boards 'above' it) through 90 degrees, or c) slide a board (including those boards 'above' it) one 'position' in any orthogonal direction.". When a board moves, there are counters that will become "hidden" and others will reappear. In the case that as a consequence of a slide or rotation, lines of the indicated length of the two players are formed, the player who made the movement wins. In any other case of forming a line of only one of the colors, the player of that color wins.

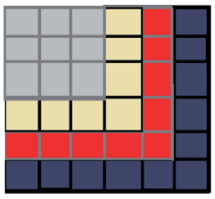


Fig. 5. Quinamid: initial board

All the sample games presented here have in common: relatively simple rules, a game board, two players alternating turns, the game does not depend on chance, and there is no hidden information at all.

III. COMPUTER SCIENCE 2 COURSE

CS2 is a second semester course in ORT Uruguay University, which continues the initial student's training in the paradigm of objects. To access this course, is a requirement to pass the Computer Science 1 (CS1) course.

CS1 is focused on teaching problem-solving methodology using OOP. The course prepares the learner for constructing simple programs using that programming paradigm. By the end of the semester, the student will be ready to analyze simple situations, to design solutions and to implement them with an OOP approach, using Java as programming language. The duration of the course is 15 weeks, organized in 4 hours of lectures and 2 hours for lab session per week. There are 25-30 students in each class. The main topics of the CS1 course are: pseudo code, variables, and control structures, objects and classes, association, inheritance, aggregation and collections, sorting and searching, and advanced use of collections.

With this background of CS1 and after completing the CS2 course, the student can analyze complex domains using methodology; design solutions and implement them under the OOP. The main topics of CS2 are: algorithm design, different data structures, tools for modeling systems based in OOP, user interface design, persistence, and exception management.

The organization of the CS2 course is similar to CS1: it lasts 15 weeks, with 4 hours of lectures and 2 hours for lab session each week. The evaluation of CS2 includes two large programming assignments in Java (done in pairs, 20 and 25 points each), participation in class (10 points) and a final written evaluation (45 points). If the student obtains 70 or more points, the course is passed. In other case, it must be retaken. Each year, we have in total 200-250 students, in groups of 25-30. Teachers of both courses have more than 5 years of experience.

The use of ASG approach has been the main axis of our CS2 course. In the next section, characteristics of the games and pedagogical approach will be presented.

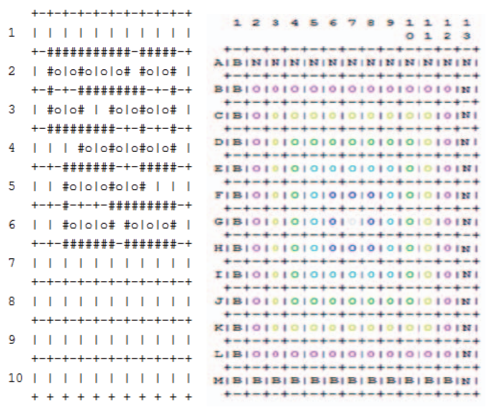
IV. GAMES IN CS2

Games have a great potential to support learning experiences and particularly game-based learning provides a better experience beyond traditional teaching approach [18].

As Tsekleves et at [19] refer, "it is yet unclear how serious games can be best incorporated within formal education systems. There is no clear pathway of how to adopt best practices" [19]. Nevertheless, games assignments are increasingly popular in Computer Science education, and particularly in Introductory Programming courses. The game first approach may be used in CS1/CS2 [10]. Drake and Sung are more explicit and refer that the major topics of CS1/2 courses can be covered with games [20]. Also, the incorporation of computer gaming into CS1 and CS2 courses, is an excellent strategy for recruiting and retaining students [21].

In particular, CS2 topics can be associated by game-related development, focus on programming skills, algorithm and data structure [22] providing a valuable educational context for students. In our case, the pedagogical approach includes the development of both assignments related to a game. The first assignment requires to implement a particular game, similar as those presented. The design of the classes requires lots of thought and some methodology should be used. Students must decide how to represent the match, where store the board, and which information belongs to the player and to the match, among other topics. They should focus on the development of an appropriated set of classes, where responsibilities are correctly assigned and modeled. The separation of domain classes and interface is promoted.

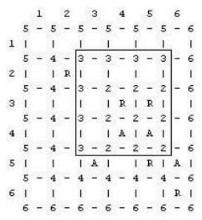
Interesting algorithms should be developed. In general, they include the use of bidimensional arrays. The interface is console and we include images of the expected representation in the proposal, to guide students. The display of the board also includes interesting algorithms to be developed. Examples of the interface of a starting match of Kulami and Morelli may look as presented in Fig. 6 and an ongoing match of Quinamid in Fig. 7.



```
    +-+-+-+-+-+-+-+-+-+-+
 1  | | | | | | | | | | |
    +-###########-#####-+
 2  | #o|o#o|o|o# #o|o# |
    +-#-+-#########-+-#-+
 3  | #o|o# | #o|o#o|o# |
    +-#########-+-#-+-#-+
 4  | | | #o|o#o|o#o|o# |
    +-+-#######-+-#####-+
 5  | | #o|o|o#o|o# | | |
    +-+-#-+-+-#########-+
 6  | | #o|o|o# #o|o|o# |
    +-+-#######-#######-+
 7  | | | | | | | | | | |
    +-+-+-+-+-+-+-+-+-+-+
 8  | | | | | | | | | | |
    +-+-+-+-+-+-+-+-+-+-+
 9  | | | | | | | | | | |
    +-+-+-+-+-+-+-+-+-+-+
 10 | | | | | | | | | | |
    + + + + + + + + + + +
```

```
   1 2 3 4 5 6 7 8 9 1 1 1 1
                     0 1 2 3
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 A|B|N|N|N|N|N|N|N|N|N|N|N|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 B|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 C|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 D|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 E|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 F|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 G|B|O|O|O|O|O| |O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 H|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 I|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 J|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 K|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 L|B|O|O|O|O|O|O|O|O|O|O|O|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
 M|B|B|B|B|B|B|B|B|B|B|B|B|N|
  +-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fig. 6. Kulami and Morelli: starting board



```
    1   2   3   4   5   6
  5 - 5 - 5 - 5 - 5 - 5 - 6
1 |   |   |   |   |   |   |
  5 - 4 - 3 - 3 - 3 - 3 - 6
2 |   | R |   |   |   |   |
  5 - 4 - 3 - 2 - 2 - 2 - 6
3 |   |   |   | R | R |   |
  5 - 4 - 3 - 2 - 2 - 2 - 6
4 |   |   |   | A | A |   |
  5 - 4 - 3 - 2 - 2 - 2 - 6
5 |   |   | A |   | R | A |
  5 - 4 - 4 - 4 - 4 - 4 - 6
6 |   |   |   |   |   | R |
  6 - 6 - 6 - 6 - 6 - 6 - 6
```

Fig. 7. Quinamid: match in course

The second assignment is an extension of the first one and includes developing a Windows-like visual interface (see Fig. 8 and Fig. 9), serialization (persistence), and few modifications or additions to the model. Some examples of those options are: to play against the computer, to give some help, to allow undoing a move, to include time-limit, and to offer the history of sequence of moves of a match.

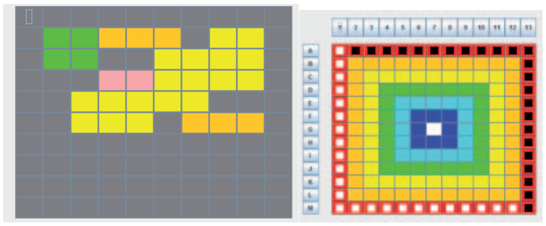


Fig. 8. Windows-like starting board in Kulami and Morelli

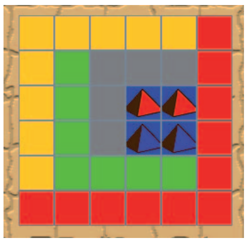
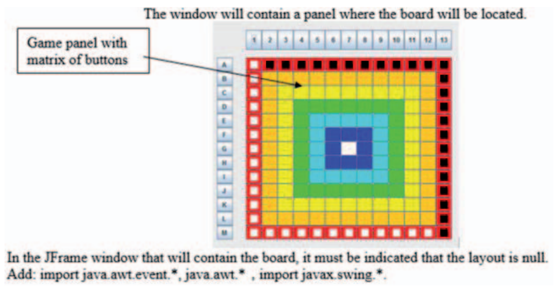


Fig. 9. Quinamid: match in course

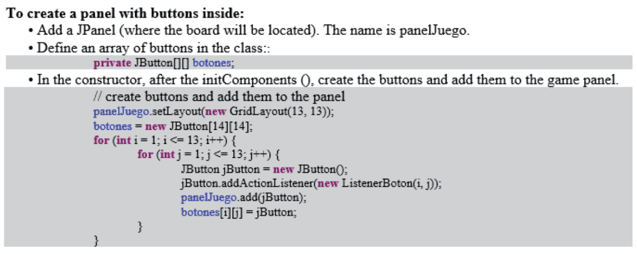
To develop the Windows-like interface, we distribute to the students a basic description of a framework with a description of how to create a grid layout in a Java Frame, with the dynamic creation of buttons used in the grid. We also provide code using Java inner classes to process which button was pressed. The use of this framework may help to understand windows design, and processing events. In Fig. 10, Fig. 11 and Fig. 12 we present a partial source code of the framework, applied to Morelli.



The window will contain a panel where the board will be located.

In the JFrame window that will contain the board, it must be indicated that the layout is null.
Add: import java.awt.event.*, java.awt.* , import javax.swing.*.

Fig. 10. Partial description of the graphical user interface framework: design.



To create a panel with buttons inside:

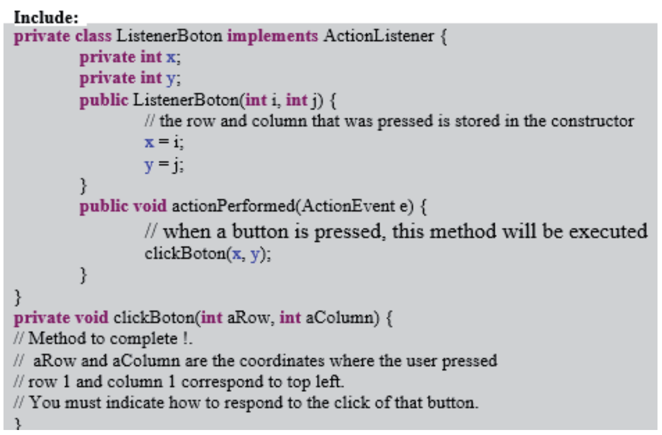
- Add a JPanel (where the board will be located). The name is panelJuego.
- Define an array of buttons in the class::

```
        private JButton[][] botones;
```

- In the constructor, after the initComponents (), create the buttons and add them to the game panel.

```
        // create buttons and add them to the panel
        panelJuego.setLayout(new GridLayout(13, 13));
        botones = new JButton[14][14];
        for (int i = 1; i <= 13; i++) {
                for (int j = 1; j <= 13; j++) {
                        JButton jButton = new JButton();
                        jButton.addActionListener(new ListenerBoton(i, j));
                        panelJuego.add(jButton);
                        botones[i][j] = jButton;
                }
        }
```

Fig. 11. Partial description of the graphical user interface framework: initial code



Include:

```
private class ListenerBoton implements ActionListener {
        private int x;
        private int y;
        public ListenerBoton(int i, int j) {
                // the row and column that was pressed is stored in the constructor
                x = i;
                y = j;
        }
        public void actionPerformed(ActionEvent e) {
                // when a button is pressed, this method will be executed
                clickBoton(x, y);
        }
}
private void clickBoton(int aRow, int aColumn) {
// Method to complete !.
//  aRow and aColumn are the coordinates where the user pressed
// row 1 and column 1 correspond to top left.
// You must indicate how to respond to the click of that button.
}
```

Fig. 12. Partial description of the graphical user interface framework: event and inner class

As a part of documentation and to promote the game, we ask to include a game description in a format similar to smartphones' app stores. Students should look for information, for instance, of graphic assets and video used to highlight and promote an application in a market [23].

To evaluate the assignments, all the teachers use an unified rubric. The rubric contains the following topics related to documentation: class diagram, coding style, testing data, and general presentation, among others. Related to program execution, the rubric includes for instance: menu, management of players, game board display, game playing, game ending, help, and persistence. For each topic, the teacher checks if it is complete and correct, partial, or not included/not developed/not ok. The evaluation includes also a "defense", which is an instance where each student should develop a specific modification of their own work in a limited time, in order to show their program knowledge and authorship. In case of a failed "defense", the student obtains 0 points for the task.

V. Course Results and Surveys

In average in the last 10 years, 80% of the students passed the course. We surveyed 3 randomly selected class groups, and obtained 54 answers of students currently attending CS2. The main results are: 81% stated that the tasks promote learning Programming (see Fig. 13); 83% referred that they are interesting (Fig. 14), and for 70% of students, the use of games is "motivating" (Fig 15).

Another results are: 76% referred that games are "appropriated", 84% referred that they are "challenging", 73% of students found correct the structure of the assignments, and 49% indicated that they had to put a lot of effort to solve them.

Some reflections given by students are: "the tasks motivate, with fun and complex proposals", "it is engaging", and "this task makes me think".

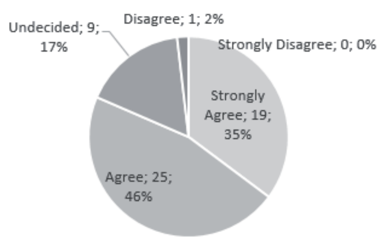


Fig. 13. "The use of games promotes learning"

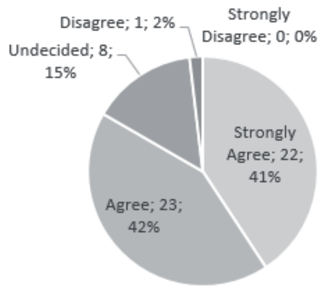


Fig. 14. "The tasks are interesting"

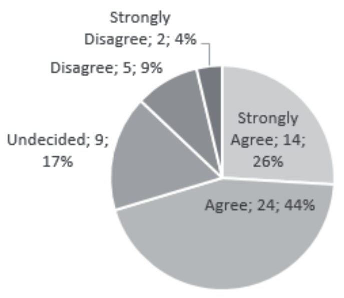


Fig. 15. "The use of games motivates"

We also surveyed 15 randomly selected students of the course of Software Engineering, a course which follows CS2. Their opinions are similar, for instance: 80% referred that the tasks are useful to learn programming, and 76% indicated that they are motivating.

Also for teachers, the use of games may "promote innovation and creativity" as refers one of them, and is "enjoyable" for other. In particular, another teacher refers that: "I find its use very important and effective. It is a way to make them to practice algorithms of a certain complexity in a domain in which they feel comfortable and motivated. As a negative aspect, I found that sometimes it is complex for them to try all the possibilities of the game, so the final programs often have several errors".

It is remarkable that students and teachers perspectives are consistent; the use of games is motivating and useful for learning programming in CS2.

VI. Lessons Learned and conclusions

Based in our experience, the selection of an appropriate abstract strategy game may take long time of searching the Internet and testing different games. After selecting one, it would be a good option that a teacher programs a full version of the game, to check the complexity of the required algorithms and prevent unexpected difficulties.

As lessons learned of using this kind of games in CS2 we could refer:

- when selecting the game, check that each match does not last too long and rules are not too complicated;
- detail carefully the game;
- show in class, if possible, an actual game created by hand to play with students;
- give some examples of particular cases to avoid misunderstandings;
- provide a required particular input and output format in order to simplify playing and testing;
- two weeks after proposing the game, ask the students to give to the teacher a preliminary class diagram, in order to promote an early engagement with the task and the opportunity to give feedback as soon as possible.

In conclusion, the topics of the CS2 course (algorithm design, data structures, user interface, persistence, etc.) can be fully covered with this kind of games. Moreover, the use of abstract strategy games in CS2, according to students' results, opinions and teacher opinions are useful and motivating to learn programming.

References

[1] Adams, R., Evangelou, D., English, L., Dias De Figueiredo, A., Mousoulides,N., Pawley, A., Schiefellite, C., Stevens, R., Svinicki, M., Trenor, J. and Wilson, D. "Multiple perspectives on engaging future engineers", Journal of Engineering Education, vol. 100, no. 1, pp. 48–88, 2011.

[2] Brayshaw, M., & Gordon, N., "Using motivation derived from computer gaming in the context of computer based instruction", SAI Computing Conference (SAI), 2016 (pp. 828-832), 2016.

[3] Bodnar, C., Anastasio, D., Enszer, J., and Burkey, D. "Engineers at play: games as teaching tools for undergraduate engineering students", Journal of Engineering Education, Vol 105, N. 1, pp 147-200, ASSE, 2016

[4] Aleven, V., Myers, E., Easterday, M., and Ogan, A., "Toward a framework for the analysis and design of educational games", Proceedings of the 2010 IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning, 2010

[5] Wassila, D., Tahar, B., "Using serious game to simplify algorithm learning". 2012 Int. Conf. on Education and e-learning innovations, 2012

[6] De Troyer, O., “Toward effective serious games”, 9th International Conference on Virtual Worlds and Games for Serious Applications, 2017

[7] Loureiro, A., Paschoal, L, Falcade, Al, and Duarte, R., “Evaluation of game-based learning approaches through digital serious games in Computer Science Higher Education: a systematic mapping. Proc. of 14th Brazilian Symposium on Computer games and digital Entertainment, 2016

[8] Buchinger, D., da Silva Hounsell, M., “Jogos Sérios Competitivo-Colaborativos.: Um Mapeamento Sistemático da Literatura”. II Cong. Brasileiro de Informática na Educação, CBIE 2013, 2013

[9] Sprint, Gina, Cook, D., "Enhancing the CS1 student experience with gamification.", 5th Integrated STEM Education Conference (ISEC), 2015 IEEE, 2015.

[10] Leutenegger, S., and Edgington, J,. “A games first approach to teaching introductory programming”, Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2007.

[11] Thompson, M., “Defining the abstract”, The Games Journal, 2000. Last accessed October 2017, http://www.thegamesjournal.com/articles/DefiningtheAbstract.shtml

[12] Heliotis, J., Bezakova, I., Strout, S. “Programming board game strategies in CS2”, Proc of FIE 2013 IEEE, 2013.

[13] “Kulami”, last accessed October 2017, http://www.steffen-spiele.de/fileadmin/Spiele/PDFs/Kulami-EN.pdf

[14] “Morelli”, last accessed October 2017, http://www.nestorgames.com/rulebooks/MORELLI_EN.pdf

[15] “Pentago”, last accessed October 2017, https://boardgamegeek.com/boardgame/19841/pentago

[16] “FlipFlop”, last accessed October 2017, http://www.nakajim.net/index.php?FlipFlop%20%28English%29

[17] “Quinamid”, last accesed October 2017, https://boardgamegeek.com/boardgame/29952/quinamid

[18] Lin, C., Huang, S., and Shih, J., “Game-Based Learning Effectiveness and Motivation Study between Competitive and Cooperative Modes”, Advanced Learning Technologies (ICALT), 2017 IEEE 17th International Conference on (pp. 123-127), 2017.

[19] Tsekleves, E., Cosmas, J., and Aggoun, A., “Benefits, barriers and guideline recommendations for the implementation of serious games in education for stakeholders and policymakers”, British Journal of Educational Technology, Vol 47, No. 1, 2016, pp 164-183

[20] Drake, P., and Sung, K., “Teaching introductory programming with popular board games“, SIGCSE, 2011.

[21] Sung, K., “Computer games and traditional CS courses”, Commun. of ACM Volume 52 Issue 12, pp 74-78, 2009.

[22] Li, Z., O'Brien, L., Flint, S., Sankaranarayana, R., “Object-oriented Sokoban solver: A serious game project for OOAD and AI education”, Procs of FIE 2014 IEEE, 2014

[23] Graphic assets, screenshots, & video, last accessed October 2017, https://support.google.com/googleplay/android-developer/answer/1078870