jOthelloT: A Java-Based Open Source Othello Framework for Artificial Intelligence Undergraduate Classes

Carlos N. Silla Jr.*, Marcelo Paglione† and Iuri G. P. Mardegan†
*Intelligent Systems Laboratory (LASIN) – Graduate Program in Computer Science (PPGIa)
Pontifical Catholic University of Paraná (PUCPR) – Curitiba, PR, Brazil 80215–901
†Federal University of Technology of Paraná (UTFPR) – Cornélio Procópio, PR, Brazil 86300-000
Email: carlos.sillajr@gmail.com, m.paglionejr@gmail.com, iurimardegan@gmail.com

***Abstract*—Introductory artificial intelligence undergraduate classes often introduce different search methods using different search algorithms. In this context one of algorithms that is often taught, is the minimax algorithm which is used in adversarial games where you want to minimize your opponent's chance of winning while maximizing your chance of winning. Different instructors use different games to make the students implement the minimax algorithm such as Checkers, Othello or Chess. However, one common problem with this assignment is that the students often spend more time implementing the game itself rather than the artificial intelligence techniques in the game. For this reason, in this paper we present a java-based open source Othello framework that was designed to be used in artificial intelligence undergraduate classes. Our framework has several features that help the students to focus on the development of the artificial intelligence aspects of the game, rather than developing the game itself. One particular feature of the framework is that it has a method that returns the list of valid moves given the current state of the game board and which player is going to make the next move. With this method, the students can focus on how to evaluate the different states using several heuristic functions and implementing the minimax algorithm. Another feature of the framework is the graphical user interface and the HumanPlayer class that allows the students to play against their own code. This feature is important as it allows the students to not only debug their codes but also to evaluate the effectiveness of their implemented heuristics. Another aspect of the framework is that it allows to set up a tournament of the codes developed by the students. The tournament can be organized in two modes. In the first mode every AI developed by one student plays against the AI developed by every other student. In the second mode, each student developed code is paired against another student developed code and only the winner plays against the winner of another pairing until there is only one winner left. An analysis of the framework in our artificial intelligence undergraduate computer engineering classes shows that it properly supports the student learning and the tournament mode also challenges them to create the best AI for Othello as they can.**

I. Introduction

The history between Othello (also known as Reversi) and Artificial Intelligence research dates back to at least the 70's, where different researchers where developing different artificial intelligence algorithms for the game. The strongest AI developed for Othello during that period was IAGO [1]. In current artificial intelligence research, Othello is being used in the development of automatically generated players using evolutionary computation techniques [2], [3], [4], [5], to evaluate existing techniques such as reinforcement learning [6] or as a test bed for novel approaches [7].

In the context of computer science education, according to Valentine [8], the game Othello has been used as a teaching tool in several areas of computer science education, such as CS1/CS2 [9], Algorithms [10], Graphics [11], AI/Machine Learning [12] and Parallel Processing [13], since at least 1979. It has also been used as test bed for comparing FPGAs and General Purpose Processors in [14].

Considering its use in Artificial Intelligence Education, Eskin and Siegel [12] have used Othello to introduce students to Genetic Programming and it is also often used (among other board games such as Checkers and/or Chess) by AI educators worldwide in minimax related assignments. One example of such assignment was presented in [10], where the students had to implement the Othello game and develop a simple AI for the game. The instructors also told the students that their assignments would participate in a tournament. This assignment was ranked highly motivating by all students of that class but a common comment from the students was that they spent more time checking for the rules of the game rather than the AI of the game itself.

The main contribution of this work is to present the jOthelloT (Java Othello Tournament) Framework[1]. In Section 2 we evaluate the impact of the framework in different offerings of our artificial intelligence undergraduate classes. In Section 3 we present some of the framework functionalities which has built in features to facilitate its use by students and instructors. In Section 4 we discuss some related work; and finally in Section 5 we presented the conclusions of this work.

II. jOthelloT in the Classroom

In this section we present a discussion on the changes on the Othello assignment based on our practical experience in six offerings of the artificial intelligence module. Table I presents the evaluation of the assignment across the different semesters.

[1]Available at: https://sites.google.com/site/carlossillajr/resources/jothellot

The symbols used in Table I are N/A for Not Applicable, ✓for full marks on that criteria and ± for some marks on that criteria.

A. First Offering

In the first offering of the artificial intelligence module in 2013-1, the students were able to team up with at most two other students and were asked to:

- Implement the Othello game;
- Implement the Minimax Algorithm with fixed-depth [15].
- Implement at least two heuristics to evaluate the non-terminal states.
- Implement at least three specialist plays.
- Design and run experiments to evaluate the different components of their developed players, e.g. which heuristics should be used? What is the impact of depth control?
- Write a detailed report explaining everything that was done and reporting any issues they had.

Furthermore, the students also had the option to decide whether or not they wanted to participate in the in-class tournament. The idea behind the tournament is to motivate the students in going beyond what was requested in the assignment.

1) Evaluation: Considering the evaluation presented in Table I, for the first offering of the module (2013-1) it became clear than having more than two students in the same team did not improve the quality of the projects. Also, most of the teams did not perform all the experiments to a satisfactory level, because some teams had issues implementing the expert system plays. Another common mistake made by the teams was to only evaluate their players in the same configuration of player 1 vs. player 2, i.e. without changing the player who begins to play the game.

2) Tournament: The different teams developed their games and respective A.I. using different programming languages. In order to run the tournament we had to have the teams sit next to each other and simulate the other teams moves as human moves against their A.I. One of the problems that some teams had was that if they clicked the wrong position by mistake, they would have to start the game from the beginning.

3) Student Feedback: All students of this module complained that they spent more time implementing the game than focusing on the artificial intelligence aspects of the game. Furthermore, two of the seven teams had issues implementing the minimax algorithm, because they were struggling with the implementation of the game itself.

4) Student Dropout: It should be noted that the team that dropped out of the class (team 7) was composed of two students from other degrees (i.e. system analysis and mechanical engineering) who choose the module as an elective.

B. Second and Third Offerings

In the second and third offerings of the module, a preliminary version of the jOthelloT framework was available. Therefore the only modification to the assignment was that the students no longer needed to implement the Othello game. Also, considering that having more than two students per team in the first offering of the module in 2013-1 did not improve the students teamwork, we have told the students to work at most in pairs.

1) Evaluation: By analyzing the evaluation presented in Table I, for the second and third offerings of the module (2013-2 and 2014-1), it can be seen that unlike in the first offering of the module, all teams who handed in the assignment were able to work on the expert plays. This suggests that the jOthelloT framework is working as intended, because since the student did not have to worry about the implementation of the Othello game, they could focus on implementing the artificial intelligence aspects of the game. However, the students were still having issues to properly design and run the experiments to evaluate their players.

2) Tournament: By using the jOthelloT framework, we were able to perform a live tournament during one of the classes. This was only possible, because all teams had to use the same programming language. In case one of the programs had an error during the tournament, the team whose program generated the error would forfeit that particular match. Another interesting fact was that in the class of 2013-2 we had a drawn between two different teams. That is the reason why Table I presents two teams as 1st on 2013-2.

3) Student Feedback: The student feedback from the live tournament was incredibly rewarding. They really enjoyed seeing each team go against each other on the big screen and found it to be highly entertaining and motivating.

4) Student Dropouts: In 2013-2 and 2014-1 we had 4 students drop-out of the module. It should be noted that 2 of these students in each semester dropped out because of the science without boarders student interchange program. The remaining drop outs were actually the same 2 students from 2013-2 who dropped out, joined the module in 2014-1 and dropped out again. One of these students is currently taking the module again in 2016-1, while the other dropped out of the computer engineering course.

C. Fourth Offering Onwards

From the fourth offering of the module, we have made some modifications to the Othello assignment. The first modification was to ask the students to implement the greedy best-first search [15]. The second modification was to raise the minimum number of heuristics to evaluate the non-terminal states that they had to implement from two to four. This was necessary as we noticed that most teams were usually implementing the same heuristics. The third modification was to alter the way the students submit their assignment. In the previous semesters, the students were requested to submit their code and report all at once. From the fourth semester onwards the students are requested to submit their code within a deadline and their report a week later.

1) Evaluation: The analysis of the evaluation presented in Table I, for the fourth, fifth and sixth offerings of the module (2014-2, 2015-1 and 2015-2 respectively), shows some interesting insights. First, considering the teams who handed

TABLE I
EVALUATION OF THE ASSIGNMENT ACROSS DIFFERENT SEMESTERS

Framework	Semester	Team	Students	Greedy	MiniMax	Heuristics	Expert Plays	Experiments	Report	Tournament Rank
Own	2013-1	Team 1	3	N/A	✓	✓		±	±	
		Team 2	2	N/A	✓	✓	✓	✓	✓	2nd
		Team 3	1	N/A	±	✓		±	±	
		Team 4	2	N/A	✓	✓	✓	✓	✓	1st
		Team 5	2	N/A	✓	✓	✓	✓	✓	3rd
		Team 6	3	N/A	±	✓	±	±	±	
		Team 7	2	N/A						
jOthelloT	2013-2	Team 1	2	N/A	✓	✓	✓	±	±	1st
		Team 2	2	N/A	✓	✓	✓	✓	✓	3rd
		Team 3	2	N/A	✓	✓	±	±	±	
		Team 4	2	N/A						
		Team 5	2	N/A	±	✓	±	±	±	
		Team 6	2	N/A	✓	✓	✓	±	±	1st
		Team 7	2	N/A						
	2014-1	Team 1	2	N/A	±	✓	✓	±	✓	3rd
		Team 2	2	N/A	±	✓	✓	±	✓	2nd
		Team 3	2	N/A	✓	✓	✓	±	✓	1st
		Team 4	2	N/A						
		Team 5	2	N/A						
		Team 6	2	N/A	±	±	±		±	
	2014-2	Team 1	2	✓		✓	✓	✓	✓	
		Team 2	2	✓		±	✓	✓	✓	2nd
		Team 3	2	✓	±	✓	✓	✓	✓	
		Team 4	2	✓	±	✓	✓	±	✓	
		Team 5	1	✓	✓	✓	✓	±	✓	3rd
		Team 6	2	✓	✓	✓	✓	✓	✓	1st
		Team 7	2	✓		✓	✓	✓	✓	
		Team 8	2							
	2015-1	Team 1	2	✓	✓	✓	✓	✓	✓	
		Team 2	2	✓	±	✓	✓	±	✓	2nd
		Team 3	2	✓	✓	✓	✓	✓	✓	
		Team 4	2	✓	✓	✓	✓	±	✓	
		Team 5	2	✓		✓	✓		±	
		Team 6	2	✓	✓	✓	✓	±	±	
		Team 7	2	✓	✓	✓	✓	✓	✓	
		Team 8	2	✓	✓	✓	✓	✓	✓	1st
		Team 9	2	✓	✓	✓	✓	✓	✓	
		Team 10	2	✓	✓	✓	±	±	✓	
		Team 11	2	✓	✓	✓	✓	±	✓	
		Team 12	2	✓	✓	✓	✓	✓	✓	
		Team 13	1	✓	✓	✓	✓	±	✓	
		Team 14	2	✓	✓	✓	✓	✓	✓	3rd
		Team 15	2	✓	✓	✓	✓	✓	✓	
		Team 16	2	✓	✓	✓	✓	±	✓	
		Team 17	2	✓		✓	✓	±	✓	
		Team 18	1							
	2015-2	Team 1	2	✓	✓	✓	✓	✓	✓	3rd
		Team 2	2	✓	✓	✓	✓	±	✓	
		Team 3	2	✓	✓	✓	✓	✓	✓	
		Team 4	1							
		Team 5	1							
		Team 6	2	✓	✓	✓	✓	±	✓	2nd
		Team 7	1	✓	✓	✓	✓	✓	✓	1st

in the assignment, all but one managed to implement at least four heuristics to evaluate the non-terminal nodes. Second, although all teams implemented the greedy best first search, there were some teams (3 in 2014-2 and 2 in 2015-1) who did not attempt to implement the minimax algorithm. Third, by changing the assignment submission from code and report to code then report the overall quality of the student reports raised significantly. Fourth, the code then report submission type has also improved the quality of the experiments designed and performed by the students. In the code plus report used in the first three offering of the modules (2013-1, 2013-2 and 2014-1) only 4 out of 14 teams (28.57%) got full marks for their experiments. In the code then report used in the following three offerings of the modules (2014-2, 2015-1, 2015-2) 16 out of 29 teams (55.17%) got full marks for their experiments.

2) Tournament: Given the success of the live tournaments in the previous semesters, we continued to run the live tournaments. However, in 2015-1, given the number of students enrolled in the class we changed the tournament format from an all against all format to a world cup inspired format. In the world cup inspired format, the teams were assigned to one of four groups (A,B,C or D). In order to assign the team to a group we had a ballot where a student from each team picked up a piece of paper with a letter corresponding to each group. The top two teams from each group moved to the knock out phase.

3) Student Feedback: As in the previous semesters, the student feedback from the live tournament was very rewarding. The students really enjoyed seeing all teams go against each other. In particular, for the 2015-1 semester, the students were even cheering and praying to not be matched up against some particular teams right of the bat. When asked about the change from code plus report to code then report, all the students reacted positively to it and said it was much a better approach as they could focus on the code first and then on carefully writing the report. Some students even asked to fix some small issues with their codes after they submitted their reports, issues that they were only able to detect because they had more time to analyse everything for the report.

4) Student Dropout: In 2014-2, 2015-1 and 2015-2 we had 5 students drop-out of the module. It should be noted that one particular student was enrolled on the module in 2015-1 and 2015-2 but never showed up to any of the classes.

III. The jOthelloT Framework

In this section we present how the jOthelloT (Java Othello Tournament) Framework can be used by undergraduate students (Section III-A) and instructors (Section III-B).

A. For Students

In order for the students to use the framework, they need to download and import the the framework in their favorite java IDE. After properly setting up the project, the students will need to create a new class which inherits (extends in java language) the `AbstractPlayer` class. By extending the `AbstractPlayer` class, the students will have to implement the constructor from the `AbstractPlayer` class and the method `play(int[][] tab)`.

The constructor from the `AbstractPlayer` class forces that the students implement the depth parameter of their minimax implementations in a standard way. This is important for the automatic tournament generation presented in Section III-B.

The signature of the method `play` is `public BoardSquare play (int[][] tab)`. This is the method that presents the students with the game board and that is called by the `Game` object when it's this player turn. It is also where the students need to develop their artificial intelligence. One method from the `Game` class that is particularly useful for the students is the `getValidMoves(int[][] tab, getMyBoardMark())`. This method returns the list of all possible and valid plays for the current player on the game board. Therefore, all the students effort go into developing the AI to evaluate the valid states and choosing their moves.

In order to test their developed players the students can either manually play against them (using the `HumanPlayerDisplay` class) or play against another automated player, such as the `RandomPlayer`. To inform the framework which classes should be loaded the students need to invoke the main method of the class `Game` and pass the players as parameters. If no parameter is informed, the `HumanPlayerDisplay` will be used against a `RandomPlayer`. We normally use this scenario as default, because we use it for the students to create familiarity with the game and test if they have successfully imported the project. Figure 1 presents the screen that should be loaded if the main method of the class `Game` is run without parameters.

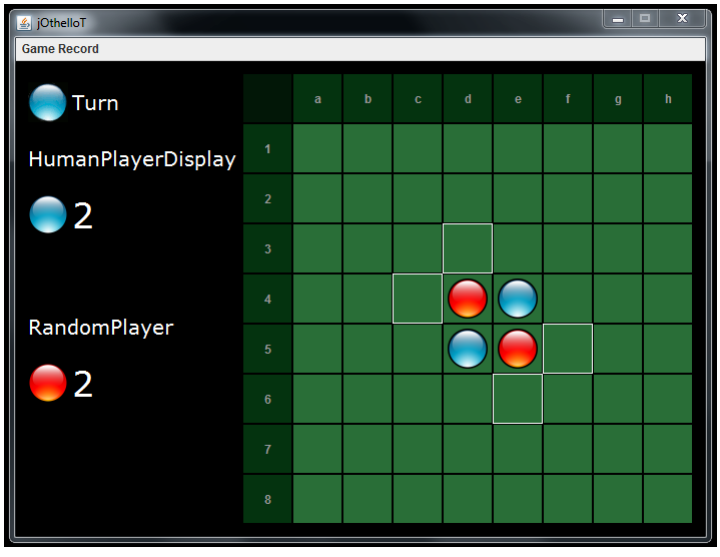

Fig. 1. The jOthelloT GUI.

B. For Instructors

From an instructor perspective, the jOthelloT Framework have some handy functionalities for the instructors to run

the live (or offline) tournament. In order to improve the use of these functionalities, we ask the students from a specific semester (e.g. 2015-1) to use the following package naming convention: `players.s2015_1.teamX` where `X` is the team number. By using this package naming convention it becomes much easier to select which of the players developed by each team should be used in the tournament. It should be noted that in our experience, each team normally submits at least two players (one using the Greedy Best First Search and the other using the minimax search).

In order to run the tournament, the instructor needs to import all the students assignments and the jOthelloT framework in his/her favorite Java IDE. After everything is setup properly, the instructor needs to invoke the main method of the class `OthelloTournament` which is found in the `tournament` package. This will load the tournament configuration menu (presented in Figure 2). In this screen the instructor can simply click which classes should be used for the tournament. Before clicking the run tournament button the instructor can also specify if the next game should be started automatically or not. Usually in our live tournament sessions, we chose to not start it automatically as some of the students like to keep track of who won/lost among some other statistics. It should be noted that all the players within a package are grouped together in this screen and this make it easier for the instructor to select the players from each team to compete in the tournament.

After each game is run, it is possible to review the each particular game summary by clicking on the `Result` button, which opens the Game Record. An example of this functionality is shown in Figure 3. The framework also automatically ranks the teams based on the tournament type. Currently there are two types of tournament available in the framework. In the first type, all teams plays against each other and get three points per win, one point per draw and zero points per loss. In the second type, the teams plays against each other on a knockout tournament.

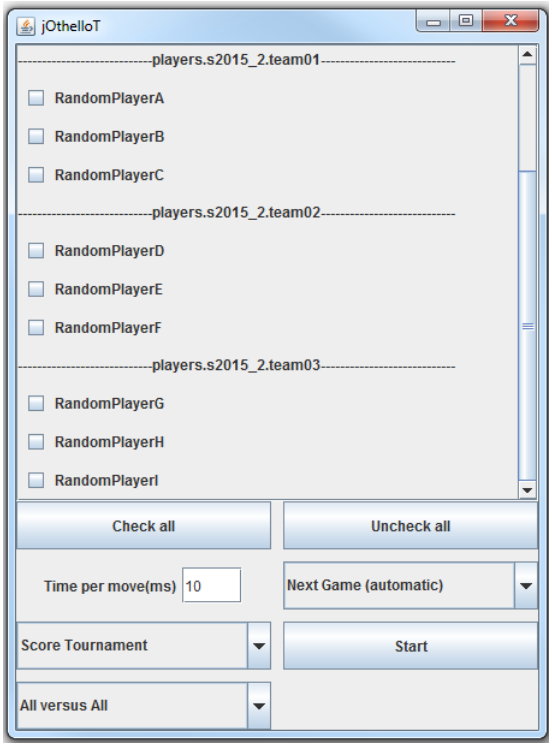

Fig. 2. The Tournament Loader GUI.

IV. Related Work

In this section we discuss some related frameworks for artificial intelligence education. DeNero and Klein [16] have created a python framework around the classic video game Pacman to teach introductory artificial intelligence concepts. Sosnowski et al. [17] created the Strategy Engine for Programming Intelligent Agents (SEPIA) which is based on real-time strategy games but was modified to support the development of the development of artificial agents rather than human play. SEPIA has flexible configuration options and was designed to be used at the undergraduate and graduate levels and well as a research testbed.

The work of El-Sheikh and Prayaga [18] present a two-phase approach for the incorporation of game-based content in the curriculum to recruit and retain computer science students. In the first phase the senior students students develop the AI and game applications for education uses by the less senior students. In the second phase the developed applications are used in various undergraduate courses. In [18] they present two outcomes of this approach. In the first outcome, one of the students developed a tutorial for the Dijkstra's algorithm with the gaming context of walking a wizard through a 2D map. In the second outcome a student developed a tutorial application for the A* algorithm in the context of the tic-tac-toe game.

Although the jOthelloT Framework has been designed to be used in artificial intelligence undergraduate classes, because it is open source it might be used for other purposes such as research testbed. Furthermore, we noticed that because it was open source, eventually some of the students from our modules would make some small contributions to it, like bug fixes and some other functionalities that were not part of the framework. When we asked the students for the motivation behind their contribution, they often replied that it was because they wanted to do something slightly different and/or improve some aspect of the framework for their use but also thinking about the students in the following semesters. In this aspect we believe that allowing students to contribute to the frameworks being used is very positive.

In the context of artificial intelligence education, the jOthelloT framework was designed for the students to focus on the development of the artificial intelligence aspects of the game without the need to implement their version of the game. In the broader context of computer science and engineering education, the game Othello and it's implementation can be used as a teaching tool to support different computer science concepts [8].

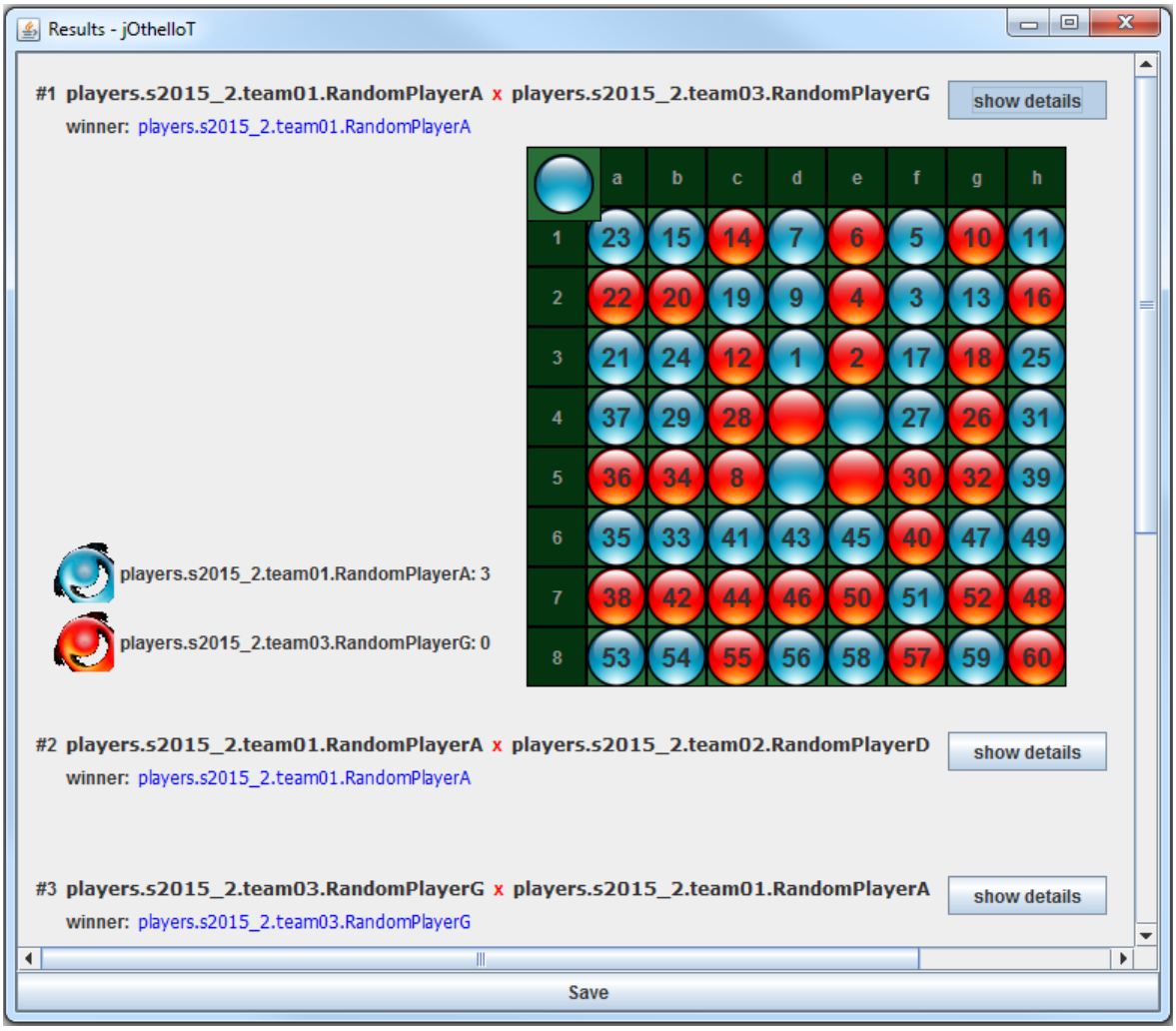

Fig. 3. The Game Record functionality that is available in the Tournament Mode.

V. Conclusion

In this paper we have presented the jOthelloT (Java Othello Tournament) framework and evaluation of its impact on different offerings of our artificial intelligence classes. Based on our evaluation we suggest that other educators who decide to use the jOthelloT framework in their classes to ask the students (in teams of at most two students) to:

- Implement the Greedy Best First Search.
- Implement the Minimax Algorithm with fixed-depth.
- Implement at least four heuristics to evaluate the non-terminal states.
- Implement at least three specialist plays.
- Design and run experiments to evaluate the different components of their developed players, e.g. which heuristics should be used? What is the impact of depth control?
- Write a detailed report explaining everything that was done and reporting any issues they had.

We also recommend that all students take part in the tournament and that the instructor use a class to run the live tournament, as our students have found the live tournament highly motivating.

Acknowledgment

The authors would like to thank all the students who took part in our artificial intelligence classes and submitted ideas, new features, bug reports and/or bug fixes. We also thank the anonymous reviewers for their valuable feedback.

References

[1] P. S. Rosenbloom, "A world-championship-level othello program," *Artificial Intelligence*, vol. 19, no. 3, pp. 279–320, 1982.

[2] K.-J. Kim and S.-B. Cho, "Evolutionary othello players boosted by opening knowledge," in *Proc. of the IEEE Congress on Evolutionary Computation*, 2006, pp. 984–991.

[3] A. Benbassat and M. Sipper, "Evomcts: Enhancing mcts-based players through genetic programming," in *Proc. of the IEEE Conf. on Computational Intelligence in Games (CIG)*, 2013, pp. 1–8.

[4] M. Szubert, W. Jaskowski, and K. Krawiec, "On scalability, generalization, and hybridization of coevolutionary learning: A case study for othello," *IEEE Transaction on Computational Intelligence and AI in Games*, vol. 5, no. 3, pp. 214–226, 2013.

[5] C. Frankland and N. Pillay, "Evolving game playing strategies for othello incorporating reinforcement learning and mobility," in *Proceedings of the 2015 Annual Research Conf. on South African Institute of Computer Scientists and Information Technologists*, 2015.

[6] N. J. van Eck and M. van Wezel, "Application of reinforcement learning to the game of othello," *Computers & Operations Research*, vol. 35, no. 6, pp. 1999–2017, 2008.

[7] T. P. Runarsson and S. M. Lucas, "Preference learning for move prediction and evaluation function approximation in othello," *IEEE Transactions on Computational Intelligence and Ai in Games*, vol. 6, no. 3, pp. 300–313, 2014.

[8] D. W. Valentine, "Playing around in the cs curriculum: reversi as a teaching tool," *Journal of Computing Sciences in Colleges*, vol. 20, no. 5, pp. 214–222, 2005.

[9] K. Becker, "Teaching with games: the minesweeper and asteroids experience," *Journal of Computing Sciences in Colleges*, vol. 17, no. 2, pp. 23–33, 2001.

[10] O. Kolas and I. Farup, "Increasing assignment motivation using a game al tournament," in *Proceedings of the 8th annual conference on Innovation and technology in computer science education*, 2003, p. 269.

[11] B. T. V. Zanden, "Optimizing toolkit-generated graphical interfaces," in *Proc. of the 7th annual ACM symposium on User interface software and technology*, 1994, pp. 157–166.

[12] E. Eskin and E. Siegel, "Genetic programming applied to othello: introducing students to machine learning research," in *Proc. of the 30th ACM SIGCSE Technical Symposium on Computer Science Education*, 1999, pp. 242–246.

[13] J.-C. Weill, "The abdada distributed minimax search algorithm," in *Proc. of the 24th ACM Annual Conf. on Computer Science*, 1996, pp. 131–138.

[14] J. Olivito, R. Gran, J. Resano, C. Gonzzlez, and E. Torres, "Performance and energy efficiency analysis of a reversi player for fpgas and general purpose processors," *Microprocessors and Microsystems*, vol. 39, no. 2, pp. 64–73, 2015.

[15] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[16] J. DeNero and D. Klein, "Teaching introductory artificial intelligence with pac-man," in *Proceedings of the Symposium on Educational Advances in Artificial Intelligence (EAAI)*, 2010.

[17] S. Sosnowski, T. Ernsberger, F. Cao, and S. Ray, "Sepia: a scalable game environment for artificial intelligence teaching and research," in *Proc. of the 27th AAAI Conf. on Artificial Intelligence*, 2013, pp. 1592–1597.

[18] E. El-Sheikh and L. Prayaga, "Development and use of ai and game applications in undergraduate computer science courses," *Journal of Computing Sciences in Colleges*, vol. 27, no. 2, pp. 114–122, 2011.