Interactive Board Games in classroom

Maria Skiadelli , *skiadelli@gmail.com*
Informatics School Teacher

Abstract

Interactive board games (IB games) are in fact board games that are played on the computer. Like traditional board games, they are played face – to – face by several human players (1-2 or more) who compete with each other following certain rules. We claim that when implemented as software products, these games can offer a richer experience to their players since they become interactive and parametrizable, features that are missing when they are implemented on a simple piece of carton. Interactive board games have one more advantage compared to other computer games: programming is necessary not only for creating such a game but also for playing it. This is the reason why we suggest a LOGO based programming environment as mostly suitable both for creating and playing IB games. In this paper we present the basic theoretical framework behind IB games, then we give an educational scenario for designing, creating and playing such games in classroom, and finally we give some examples of IB games implemented by students of the 6th Grade of primary school.

Keywords

board games, logo, programming, informatics curriculum

Introduction

For many years children and adults used to have a lot of fun by playing board games on a simple piece of carton. Nowadays computer games have dominated their free time, so that board games have been rather put aside. Computer and video games are much more attractive since they are colourful, parameterizable, offer action and interactivity. However it seems that board games are still important for many people, since they enhance strategic skills, rule based gaming and face – to – face interaction between the players [Kafai Y. 2001; Retalis S. 2008;]. Moreover the randomness, on which many of them are based, is an undisputable fun factor. Therefore trying to design and implement board games that can be played on the computer by one or more players can lead to a next generation of board games, which we call interactive board games (IB games). We chose this name since we believe that the main feature that is added to their rather static nature by transferring them on the computer is interactivity.

Interactive board games and computer games: a comparison

We will try to give a simple definition of a board game by quoting Wikipedia's relative article found in http://en.wikipedia.org/wiki/Board_game:

*[A **board game** is a game that involves counters or pieces moved or placed on a pre-marked surface or "board", according to a set of rules. Games can be based on pure strategy, chance (e.g. rolling dice) or a mixture of the two, and usually have a goal which a player aims to achieve. Early board games represented a battle between two armies, and most current board games are still based on defeating opposing players in terms of counters, winning position or accrual of points (often expressed as in-game currency).*

There are many different types and styles of board games. Their representation of real-life situations can range from having no inherent theme, as with checkers, to having a specific theme and narrative, as with Cluedo. Rules can range from the very simple, as in tic-tac-toe, to those describing a game universe in great detail, as in Dungeons & Dragons (although most of the latter are role-playing games where the board is secondary to the game, helping to visualize the game scenario).]

The definition of a computer or video game comes also from Wikipedia found in http://en.wikipedia.org/wiki/Computer_game:

*[A **video game, computer game or console game** is an electronic game that involves human interaction with a user interface to generate visual feedback on a video device.]*

Here we should notice the fact that the IB games that we present in this paper should be also considered as computer games, since they are played on the computer. However they do not fall into the usual computer game categories: action, adventure, role-playing, simulation and their combinations [Zagal et al. 2006]. We would rather say that they form a new category of computer games. However for simplicity reasons, we will keep on using the term **computer game** to refer to the usual video, computer, console and electronic games, whereas we will keep the term **interactive board games (IB games)** to refer to board games implemented on a computer, although the distinction sometimes can be vague.

Computer games are often complex software products and their underlying mechanisms are frequently opaque to the average player. In contrast, board games are simple. Their game play is fairly constrained and their core mechanisms are transparent enough to understand. Besides most computer games are individual, while board games are multiplayer by nature and require face to face communication.

Another difference between IB games and computer games is who creates the action. In an IB game, the action of the game is created by the human players of the game supported by the chance and of course the rules of the game, whereas in a computer game the action is mainly created by the computer and the humans (one or more) just respond to this action or series of events. This leads us to think that IB games help their players to take the initiative over the machine and develop less pathetic behaviors towards the computer.

Last but not least, computer games are difficult to implement; they need a lot of effort, time and programming experience of a superior level. Often the results are not rewarding for young children and novice programmers since the games that they create are simplistic and not satisfactory enough for playing. Especially children that have the experience of professionally designed and implemented games can be easily disappointed by the result of their efforts. Any game should be: enjoyable, different each time and should have a satisfying result.

These are the reasons why we believe that board games worth being re-invented and continue to be played by children and adults.

Building IB Games in classroom: collaborative activities based on the constructionism learning theory

The advantages of using board games in education are well documented in literature: "*A board game is played by multiple players who move pieces across a premarked surface using counters or dice. Adding board games to the educational process can lead to an interactive learning experience*" (Helliar et al., 2000).

As it is obvious, the idea of transforming a traditional board game to an electronic one is not a new one (Retalis S., 2008). What bears some originality here is that we suggest that this has to be done by children or novice programmers in an educational activity framework.

The idea of games created by the children for educational purposes is a rather popular and well known constructionist idea. Y. Kafai in her article "Playing and Making Games for Learning" mentions (Kafai Y., 2006): *"The instructionists, accustomed to thinking in terms of making instructional educational materials, turn naturally to the concept of designing instructional games. Far fewer people have sought to turn the tables: by making games for learning instead of playing games for learning. Rather than embedding "lessons" directly in games, constructionists have focused their efforts on providing students with greater opportunities to construct their own games—and to construct new relationships with knowledge in the process".*

Therefore, designing and building of IB games is clearly a constructionist approach fully influenced by the constructionism learning theory (Kafai Y. et al., 1996). To think of Seymour Papert's popular saying: "Constructionism shares constructivism's connotation of learning as "building knowledge structures", irrespective of the circumstances of the learning. It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity".

Besides, the face to face way of playing, makes a board game suitable for collaborative classroom activity (Zagal J, 2006, . Groups of students can collaborate not only at the design and building phase of the games but also while playing them. In classroom collaboration and interaction can really be leveraged by activities based on IB games.

IB Game making in the Informatics curriculum

At the initial stages of teaching programming, one has to find meaningful activities that are easy to be implemented by the learners in short time. We claim that the development of simple interactive board games fall into this category of activities. We also claim that a logo based programming environment is mostly suitable for creating such games. The LOGO language as it is well known possesses the inherent feature of drawing lines as traces left by the turtle objects (Papert S., 1993). Therefore it is rather easy to use such an environment for programmatically creating the board of the game, instead of doing so by using a common drawing tool. Such a practice has the following advantage: the board of the game may be changeable rather than being static. A changeable board may produce various gaming situations each time the game is played, or if their players want to do so. And it is not only the board but also the "dice" - or in more general terms the element that controls the randomness factor – that can be also created programmatically.

While making a game, a learner has to take a number of design decisions while he/she starts developing technological fluency. Just as fluency in language means much more than knowing facts about the language, technological fluency involves not only knowing how to use new technological tools but also knowing how to make things of significance with those tools and most important, develop new ways of thinking based on use of those tools. Beyond that, game-making activities offer an entry point for young gamers into the digital culture not just as consumers but also as producers (Kafai Y., 2001).

Due to the interpreted nature of the LOGO language (Papert S.: 1980) the programming environment can be used not only for programmatically creating the game, but also for playing it by using programing instructions of the LOGO language. The movement of each piece on the

board can be in fact realized as the movement of a turtle on the LOGO canvas. Of course a turtle can move only when it receives the right instructions. That is something that helps learners to get familiar with the basic repertoire of instructions of the programming language and use the instructions to do something useful and meaningful i.e. to play the game. Moreover, in a simpler educational scenario intended for early childhood children, kids could just play a game created by someone more experienced in programming for instance their teacher or older children, by just giving the right programming instructions to the turtles on the electronic board.

The IB game educational scenario

As we mentioned earlier what interests us from the educational point of view is how to build and/or play IB games in classroom. In this paragraph we present an exemplar educational scenario (Beetham, H., 2007) that summarizes how this can be done. We chose to write the scenario in the form of a learner's activity worksheet, so that it can be easily understood and used by other teachers. This scenario is mainly meant to address groups of children between 11-13 years old, but it can also be used with groups of novice programmers or teachers with limited programming experience. The goal of the scenario is to guide the learners to create their own game, thinking first about design issues such as the rules, the narrative, the subject, etc. and then try to program their game. The scenario is divided in 3 stages. Each stage follows certain steps.

1st stage: designing the game

1. Think of a board game category. Board games fall into plenty of categories: they can be path games that are purely based on luck and chance like Chutes and Ladders or they can be based on strategy like Tic-Tac-Toe.
2. Whatever the category, it's better to find a theme. The theme can be related to a fairy tale, a book, some everyday life situation, etc. Pirates, kings and castles, space and horror, usually inspire children, although for adults, the scenarios can be different.
3. Map out the rules and the directions of the game. Some questions that a game designer must pose to himself/herself are the following:
   - What is the end goal of the game?
   - How would the players win?
   - What is the minimum and maximum number of players that can play?
   - What each player has to do every time it's his/her turn to play?
   - What are the pieces needed for the game? (Players' markers, dice, cards, etc.)
4. Sketch a rough draft of your board design on a piece of paper. This will allow you to determine whether you need to include more or less details in your final design. For path games, one needs to add start and finishing places, and set out a clear path or road for the character(s) to travel along.

2nd stage: implementation

5. You can choose to create your own designs for the images and pictures that will go on your game, but if you would rather use ready-made images, there are many resources on the Internet that you can do a search on and download to the theme selected in step 2.
6. Import any graphical images found and chose in step 5
7. Start to build the game programmatically. Create one procedure that builds the board of the game (that may call of course several sub-procedures).

8. Think of how you will treat randomness if the game is based on luck, i.e. what will serve as your dice and how the dice can be programmatically implemented.
9. Write one procedure that does game initialization (placing board, players and dice at original positions).

3rd stage: testing and playing

10. Test and retest your game plenty of times, by playing alone or with some peers. By testing it you can correct any unforeseen bugs or pitfalls. Ensure that the game rules are fair and that the game concept is fun and exciting for the target audience.
11. Ask from another group of peers to play your game, and try to play theirs. Discuss your experience and try to make some comments to help them improve their game. Make improvements to your game according to their comments.
12. Try to write a small manual of instructions, to help other people to play your game.

The above educational scenario is basically proposed for teaching initial algorithmic concepts and basic programming skills to children and novice programmers as said earlier, so it can be part of an informatics teaching curriculum.

The advantages that occur from the implementation of such a scenario in classroom can be summarized as follows:

- Children develop creativity and imagination since they are urged to think of the category and the theme of the game, the design of their characters the role playing scenario that defines the action that takes place while playing the game. Aesthetic criteria decision making and self-esteem are strongly enhanced at this stage.
- They have to come up with rules and directions: who wins, what a player should do when he/she takes turn. Implementing rules helps children develop logical thinking and strategic skills.
- While implementing the game by programming, they also develop basic programming skills and learn to program the computer by doing something that is meaningful.
- The last step of the scenario supports good-fellowship and camaraderie between children. Interactive board games like common board games need usually at least two players that play face-to-face. Interaction takes place between humans without the mediation of the machine.

In the next paragraph we will give some examples of IB games that were implemented in classroom by students of the 6th grade of the 10th public primary school of Maroussi in the Athens area during school year 2010-2011. The children worked in groups under the supervision of the ICT teacher for a period of two months (about 16 teaching hours) following the proposed educational scenario.

Examples

Example 1: the coloured dots game

The examples of the IB games presented in this section have been implemented with EasyLogo. EasyLogo is a LOGO based programming environment for young children or adults that want to acquire basic programming skills. (Salanci L., 2010). Easy Logo is a user friendly and easy to use programming environment with a very limited set of commands (~10). However due, to its simplicity, we consider it as the right programming environment for young children. The only serious deficiency that we noticed during the implementations of the games was the fact that

there is only one turtle object on each canvas.

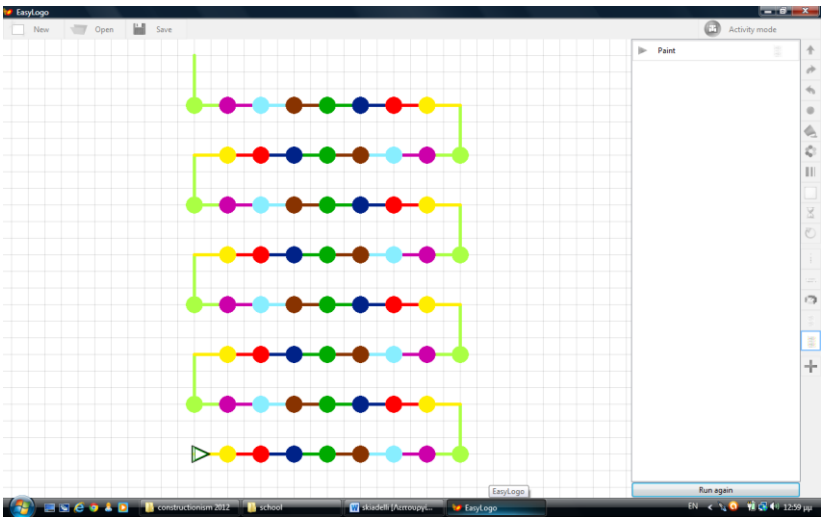

Figure 1. The coloured dots IB game

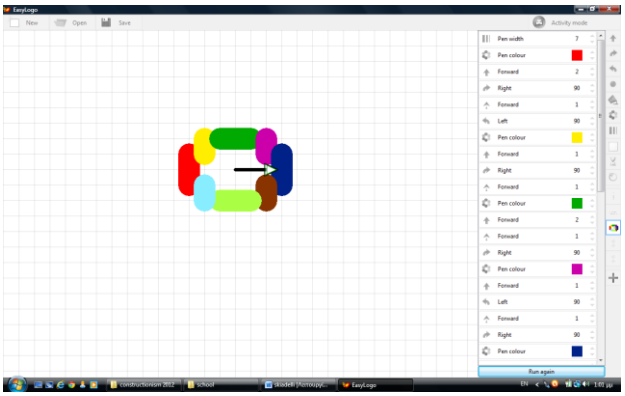

Figure 2. The implementation of the dice

The rules of this game are the following: players play in turns. Each player needs to roll the dice each time that comes his/her turn. To roll the dice the player presses the "Run Again" button and the arrow moves to a random colour of the circle. Then he/she gives the right logo commands in order to move the black arrow that appears on the board (i.e. the turtle) to the first dot that has the colour indicated by the arrow of the dice. The player that first reaches the last dot wins.

This game is a game based on luck that needs two or more players. It's a very simple game that can be played even by very young children to help them practice basic LOGO language commands, orientation and numbering skills.

Part of the instructions given by the players for playing the game appears below:

paint[1] *3*
forward 6

[1] The command "Paint" in EasyLogo, in fact means "Run the procedure". The number argument that follows this command is the code number of the procedure as assigned by the system each time. So "Paint 3" means "Run the procedure No 3" which is the procedure that draws the canvas of the game and does the initialization.

forward 4
forward 6
left 90
forward 3
left 90
forward 8

The game is implemented by four procedures: two for creating the board, one for creating the dice and one for the initialization that calls the other three. The first command given by the first player in the programming pane calls the fourth procedure. Below that call, each player places the programming instructions that move the arrow – turtle on the game board. The players have to change between the dice and the playing space canvas.

There can be a lot of alternative implementations both of the board and of the dice of this game. Another kind of board and dice that could be used for this game, are shown below:

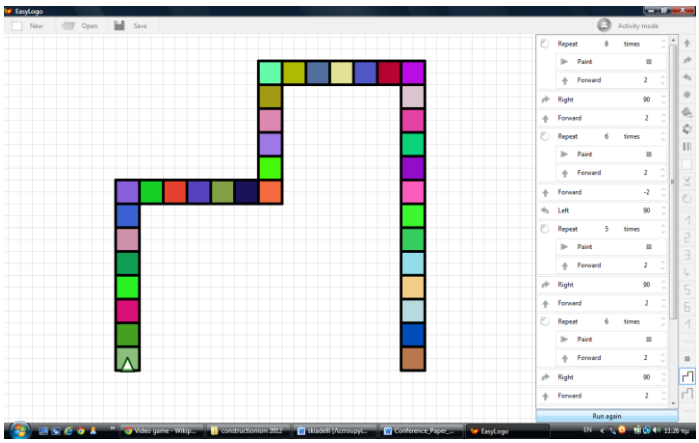

Figure 3. The coloured squares game (2)

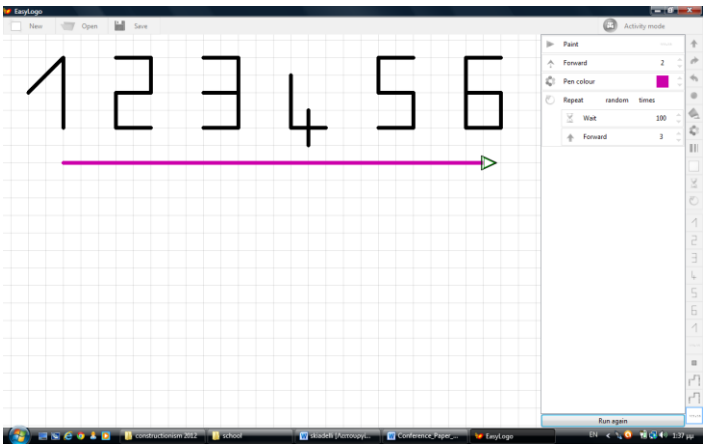

Figure 4. The implementation of the dice (2)

Example 2: the tic-tac-toe game

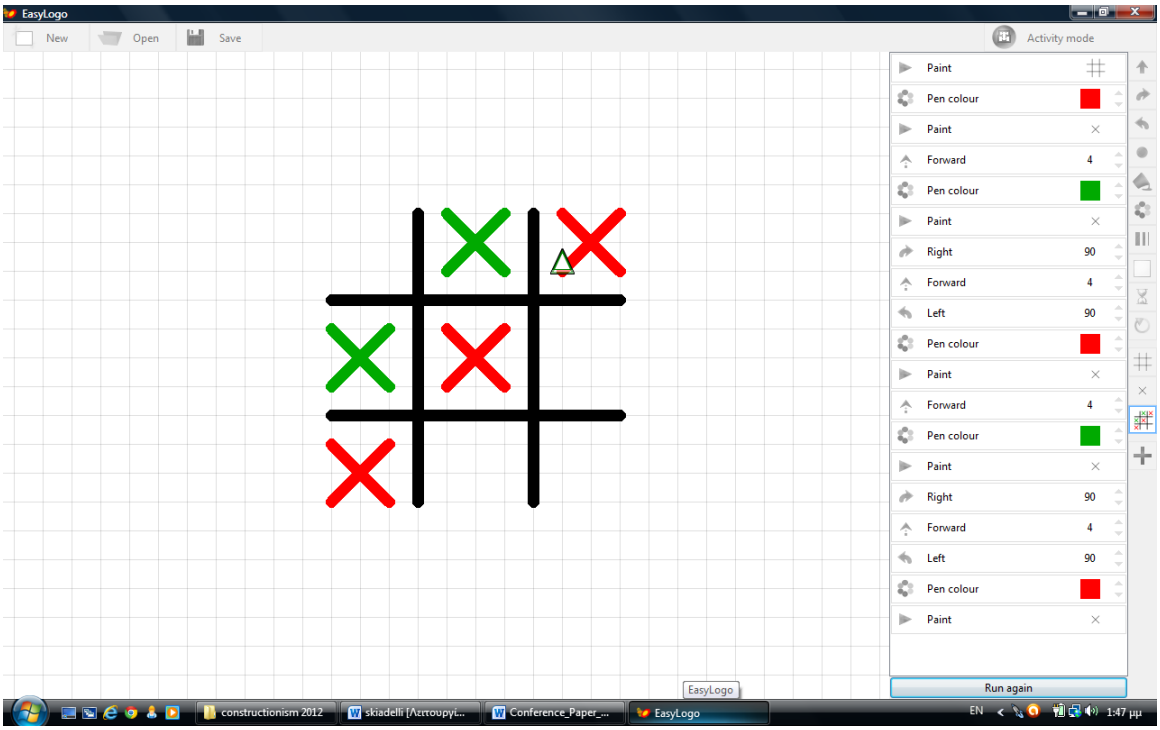

Figure 5. The tic-tac-toe game

This is the very common tic-tac-toe game that is played in the normal way by two players by giving programming instructions to place their X's on the cross board. The game is built by two procedures, one that builds the board, and the second that builds the **X** symbol. Each player has to put the X to the right place of the board by first moving to the right position and then make a procedure call to the **X** symbol procedure. Before that he/she has to change the colour of the pen to one that corresponds to his/her colour. A part of the code written by the two players while playing follows:

paint 0
pen colour = red
paint 1
forward 4 with pen up
pen colour = green
paint 1
right 90
forward 4 with pen up
left 90
pen colour = red
paint 1
forward 4 with pen up
pen colour = green
paint 1
right 90

forward 4 with pen up
left 90
pen colour = red
paint 1

Alternatively, the players can use the X symbol procedure and the dot drawing command, instead of the two different coloured X symbols.

Example 3: the maze

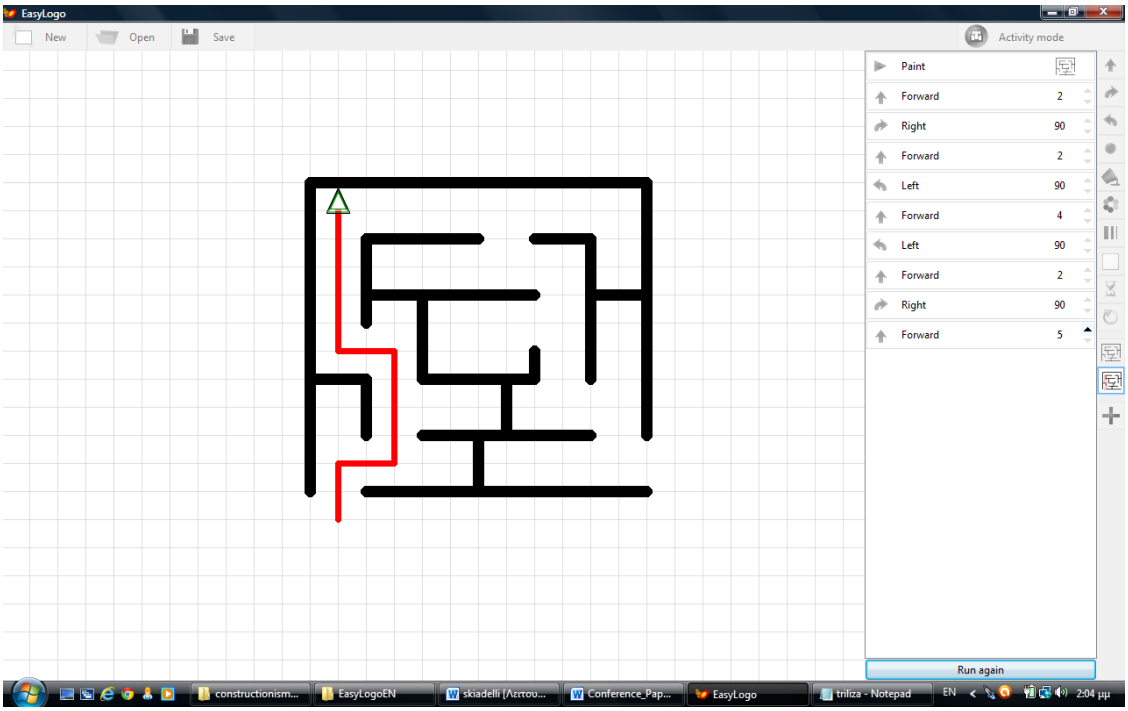

Figure 6. The maze game

The maze game is played by two players. One player builds the game and the other has to get out of it. One can also put a time restriction to make the game more challenging. What makes this implementation interesting is that the maze is drawn programmatically and therefore each time can be drawn in a different way. Again the player that has to get out of the maze needs to do so by giving programming instructions.

Conclusions and future ideas

The IB game scenario offer an educational framework for children (11-13 years old) and novice programmers to create simple but still meaningful software products that can share with their peers. Due to some of the inherent features of the LOGO language, these games can not only be created programmatically but also be played programmatically even by very young children between 6 and 9 years old. There is a lot of creativity that comes up with such kind of educational activities: children have to think of the category and the theme of the game, come up with rules and directions that have to be followed, develop strategic skills or other techniques to master the game. In the future it would be nice to explore the possibility to play such games on a touch screen environment like tablet pc or smart phone, interactive table board (MS) or even a touch

floor. The idea of how the LOGO based environments can profit from touch based or movement based interface technologies has not yet adequately explored. This would give new perspectives to these environments and therefore to the interactive board games that can be built with them.

References

Beetham, H. (2007): An approach to learning activity design. In Beetham, H. and Sharpe, R. (eds.) *Rethinking Pedagogy for a Digital Age* pp. 26-40, Oxford, RoutledgeFalmer.

Helliar, C.V., Michaelson, R., Power, D.M., & Sinclair, C.D. (2000). *Using a portfolio management game (Finesse) to teach finance*. Accounting Education, 9(1), 37-51.

Kafai, Y. & Resnick, M. (1996*): Constructionism in practice: Designing, thinking, and learning in a digital world*. Mahwah, NJ: Lawrence Erlbaum Associates (eds.).

Kafai, Y. (2001): The Educational Potential of Electronic Games: From Games–To–Teach to Games–To–Learn. *Playing by the Rules Cultural Policy Center, University of Chicago* October 27, 2001 retrieved by http://culturalpolicy.uchicago.edu/papers/2001-video-games/kafai.html

Kafai, Y. (2006): Playing and Making Games for Learning: Instructionist and constructionist perspectives for game studies. *Games and Culture*, 1(1)., pp. 36-40

Papert, S. (1980) *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York.

Papert, S. (1993): *The Children's machine, Rethinking School in the Age of the Computer*. Basic Books, New York.

Retalis, S. (2008): Creating Adaptive e-Learning Board Games for School Settings Using the ELG Environment. *Journal of Universal Computer Science*, 14 (17), 2897-2908

Salanci, L. (2010): EasyLogo – discovering basic programming concepts in a constructive manner. In: *Constructionism 2010, European Logo Conference*, Paris.

Zagal, J., Rick, J., Hsi, I. (2006): Collaborative games: Lessons learned from board games. *Simulation & Gaming*, 37(1). (March 2006), pp. 24-40