# Web based Adaptive Educational Games - Exploitation in Computer Science Education

Konstantinos Maragos *

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications

kmaragos@di.uoa.gr

**Abstract.** In the context of this dissertation, issues related to design and development of educational games that support the teaching of programming are studied. Specifically, the dissertation is placed in the context of supporting the learning process by utilizing the Web and especially Web based, Adaptive Educational Games. A Web based Adaptive and Multiplayer Educational Game TALENT (Teaching ALgorithms EnvironmeNT) was designed and implemented, which motivates and supports students in learning programming.

**Keywords:** Programming, Mini-Languages, Adaptive Systems, Educational Computer Games, Computer Science Education

## 1 Introduction

Much of the existing research, in the computing education literature at least, focuses on new and interesting ways to teach programming [1]. One way is to focus on game based learning, which can be defined as the use of a computer game based approach to deliver, support, and enhance teaching, learning, assessment and evaluation [2]. Computer games are well suited to use within an educational environment because they build on theories of motivation, constructivism, situated learning, cognitive apprenticeship, problem-based learning and learning by doing [3]. Furthermore, educational computer games improve intrinsic motivation through challenge, curiosity, control and fantasy [4].

On the other hand educators have to get more attention on the fact that today high school students have been brought up in a technologically rich environment. They belong to the generation called “digital natives” or “net generation” [5]. These students were "born digital" and they have been heavily influenced by the latest highly interactive and individual technologies such as iPods, iPhones, iPads, Wii, Xbox, Playstation game consoles, as well as Wi-Fi Internet access and graphic rich multiplayer Internet gaming. In addition, as result of the Internet, students learn much more collaboratively today than in previous generations. Thus, there is an important need

*Dissertation advisor: Maria Grigoriadou, Emeritus Professor

for educators to embrace and adopt approaches to teaching and learning that are better suited to the learning styles that the younger generation of learners now adopt and provide a more stimulating and engaging learning environment.

According to [6], the most common method of introducing students to programming is the gradual presentation of programming structures for a general purpose computer language and also, the presentation of increasing difficulty problems, based on these structures. This approach is inadequate for students, as it provides them a series of obstacles [7]. This is augmented by the fact that students today seem to be motivated by graphically rich learning environments that have multimedia characteristics [8].

An alternative and more effective approach for introducing students to programming is to provide them with mini environments, based on micro worlds and mini languages [1], [9]. According to this approach, students learn to program by instructing a virtual entity (e.g. turtle, robot) in a virtual micro world. This virtual entity can be controlled by a small set of instructions and basic programming structures like. conditions and loops. There are several examples of games that provide interaction through a mini environment and a mini language like Robocode (2000), Gun Tactyx (2004), Ceebot (2005), Robomind (2005), Sheep (2006), Rapunsel (2007), LightBot (2008) and Marvin's Arena. The main issues of these educational games are the type and the story of the game, the programming language they are using, the method of programming, the capabilities of the execution of the program, the ability to support many users at the same time that are able to communicate with each other and the exploitation of the web technologies.

In addition, all the above games they do not take into account the student model which provides an additional resource of the learning process that is believed to enhance the learning [10], [11]. The student model is a representation of understanding, difficulties and misconceptions of the student [12] enabling the system to adapt to student needs and current learning requirements [13].

In the context of supporting programming education the design of a web based multiplayer adaptive educational game is proposed, referred to as TALENT (Teaching ALgorithms EnvironmeNT) [14], [15], [16], [17], which (i) is based on a story-context and the activities are grouped according to learning goals, (ii) uses specially designed mini language that supports all programming structures: sequence, selection and repetition, (iii) utilizes visual programming techniques to create programs with no limitation on the number of program's instructions, (iv) allows the programming of variables corresponding to the properties of objects transforming the virtual world from static to dynamic, (v) allows a step by step execution of program's instructions, (vi) creates and utilizes the model of student providing adaptation, (vii) exploits the potential of the internet, does not require any installation, supports the simultaneous presence and interaction of many users and the synchronous communication between them as well and (vii) includes the design of an authoring tool.

## 2 The TALENT Environment

TALENT is designed as an adventure multiplayer web based educational game for teaching programming. Adventure games offer powerful opportunities for learning and development of problem-solving abilities [18]. Adventure games are also suitable for science concepts that may be hard to visualize or manipulate with concrete materials [19] and programming is a kind of such a concept. In addition, adventure games, which are consisted of tasks, provide a clear progression overall in the game [20].

### 2.1 Programming in the game

In the educational game TALENT, when a student is engaged in a programming activity, they have to use the mini language to instruct a robot vehicle to do the necessary actions in order to succeed in mission in the microworld. More complex missions require more complicated programming. For this reason the student in TALENT is able to construct more advanced programs by using programming structures (e.g. selection, repetition). The student is able to construct the programs in the environment with the support of a prototype online programming tool (fig. 1) which combines a program editor and an interpreter running on the web. The online program editor permits students to drag instructions, represented by buttons, from the instruction toolbox and drop them on programming area. The student's program is formed by this drag and drop of several instructions and programming structures.

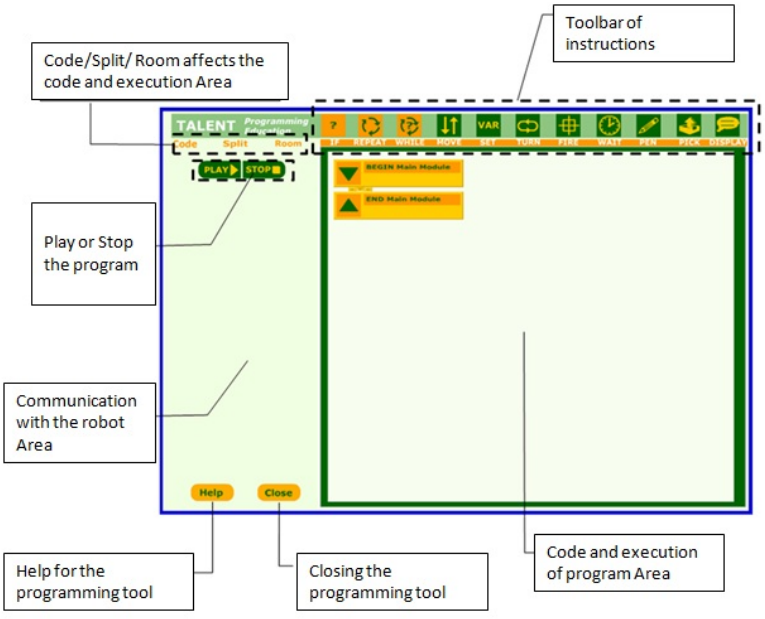


**Fig. 1.** The programming tool of the Game

When the program is ready, the student can go on with execution using the embedded interpreter. On program execution, the environment maintains both the micro world and the student's program, visible on the screen. The program executes one instruction at a time, while the interpreter highlights programming constructs in the source code as they are being executed and the effect is simultaneously shown in the micro world. This makes the connection between a mini language instruction or construct and its effect on the micro world obvious. The programming tool has no limitation about the number of the instructions, a drawback found in other games.

### 2.2 Student Modeling and Adaptation

The first step to create an adaptive educational game environment is to create several different activities, indicate for each the learning goals involved and then group these activities into activities groups [21]. Activity is the basic unit of the game structure and indicates a programming task to be performed. Following this methodology we divided the whole game world into three tiers Map, Place and Mission. The Map tier is the general map of the game world that is presented to the student when the game starts and outlines the places that a student can visit. A Place is the part of the game which outlines the missions that are proposed to the student. Finally, Mission corresponds to a programming activity presented to the student. Each programming activity has its own learning goals. There exist many learning goals concerning programming concepts like the use of variables, selection and repetition. Finally, a set of programming activities with the same learning goal is grouped together forming the Activity Groups. The number of the different activity groups is equal to the number of the different learning goals.
The first time the student enters the game, he has to register. During registration the student gives his own personal characteristics (age, gender etc,), chooses a nickname, which must be unique, and builds the presence of his own avatar so to be distinguished by the other students. These data records are saved in the game database and are associated with the student model.

The TALENT game gathers information about learner's navigation, the use of tools, the progress in programming activities and the achievement of learning goals. In addition, data that concerns the sequence of places and programming activity selection, the total time spent on a place or a programming activity, the total number of times that the student asked for help in a programming activity, the visits to a programming activity or a place, the number of times he communicated with the other students in the game and the progress on learning goals are saved in a separate database record according to student's nickname. The system uses all of these data records to build the student model and to adapt on students abilities with curriculum sequencing and navigation support. This means that system proposes the next mission to the student and also helps the student to navigate to this mission. Student model is visible by the student providing student with power over his learning [22]. Furthermore, one can compare the information in his model with that produced with the

mean data of all the other students that is argued to motivate student to try to succeed more in the activities [23].

## 3 The Empirical studies

For the evaluation of the Educational Game TALENT, two empirical studies were carried out, the first with university students and the second with high school students. In the following subsections we are going to present the procedures and the results of the two studies.

### 3.1 The 1st empirical study - Use of TALENT by University Students

The first empirical study was conducted during the lesson "Informatics and Education" of the Informatics and Telecommunications Department of the University of Athens. The participants of the research were 122 four year graduate university students of the lesson. The students were already taught concepts related to ICT in Education, educational software, educational games, educational programs, simulations, programming languages, educational resources on the web, educational case studies, distance learning, the role of the teacher and evaluation of educational environments.

The evaluation of TALENT was given as one of the written exercises of the lesson. Students were asked to use the programming environment of the game and then to rate the system's usability, to write down advantages and disadvantages, and finally to propose didactic scenarios to be embedded in the game. For the rating of the system usability the Computer System Usability Questionnaire (CSUQ) [24] was used. The questionnaire includes 19 questions about system usability in a 7-point Likert scale [25] (7=strongly agree, 1=strongly disagree).

Furthermore, in order to collect the students' opinions, comments and suggestions about the operation of system components a second questionnaire with open type questions was used. Indicative questions of the second questionnaire were "what do you think about the usefulness of TALENT in the process of teaching elementary programming concepts", "summarize the advantages and disadvantages of the use of TALENT", and "what was your experience from the design of didactical scenarios with TALENT's mini language".

The research lasted about a month and the procedure of the research consisted of the following three phases:

Phase 1: Prior the evaluation, the students participated in a four hour class lesson. The lesson was divided in one hour lecture and three hours in the computer laboratory. During the one hour lecture several subjects about educational computer games, design issues and results from game based learning research was discussed. During the three hour lesson in computer laboratory several paradigms of educational computer games especially for computer programming were presented, as well as the syntax and the educational tool of the mini language of the TALENT.

Phase 2: After the four hours lesson the university students had to deal with an exercise which was the use of the game mini language in given didactic scenarios about

the three programming structures, sequence, selection, repetition. Specifically, students had to program some activities using the mini language tool of the game.

Phase 3: After the use of the programming tool of the game, students had to answer the two questionnaires, the Computer System Usability Questionnaire and the questionnaire with open type questions about the advantages and disadvantages of the system as an educational tool. Finally they had to propose a programming activity that could be embedded in the game.

### 3.2 The 2nd Empirical study - Use of TALENT in Classroom settings

The second empirical research was conducted in the curriculum of the lesson "Informatics Applications" in a high school of Athens, Greece. Overall, 65 students (n=65), 33 females and 32 males participated in the research. Students were at the age of 15-16 and were already familiar with programming, as they were taught programming languages from the junior high school. The research was carried out in the computer laboratory of the high school and the duration of the research was 8 weeks. At the beginning, all the students participated in a pre-test. After the pre-test, students were randomly assigned to one of two groups, the control group and the experimental group. There was no significant difference found on the t-test performed between the two groups according to the pre-test performance (t=-0.255, df=63, 2-tailed p=0.799).

The concept of "if-then-else" and "while loops" in informatics teaching was used as the experimental content. At the end, all the students participated in a post test. The procedure of the 8 weeks research consisted of the following five phases:

Phase 1: Introduction to TALENT game environment (1st and 2nd week) – lasted 4 hours. During the first phase the components of the game was presented to students (2 hours) and all the students were engaged in programming tasks about the concept of variables (2 hours).

Phase 2: All the students completed the Computer System Usability Questionnaire (CSUQ), like the university students, about the system usability (3rd week) – lasted 1 hour.

Phase 3: Pre-test (3rd week) - lasted 1 hour. All the students took the pre-test consisted of questions and exercises about "if-then-else" and "while loops".

Phase 4: Working in groups (4th-5th-6th-7th week) - lasted 8 hours. Experimental group studied the concepts of research using only the TALENT environment. The students used the game and mini language tool of the game to program the game activities. An example of such a programming activity for the concept of selection control structure is shown in figure 3. The role of the teacher was limited on the support of system operations, when needed. On the other hand, the control group participated in a traditional lesson where the teacher carried out lectures about the main concepts of the research. During the traditional lesson, the teacher made questions in order to address the prior knowledge, explained the concepts, answered queries and solved exercises.

Phase 5: Post-test (8th week) - lasted 1 hour. All the students from the experimental and control group participated to a final written test.

The research tools were the questionnaire, the log files created by the students' actions in the game and finally the two tests, pre-test and post-test. The questionnaire was based on Computer System Usability Questionnaire (CSUQ) extended with three open questions in order to extract students' opinions about the advantages and disadvantages of the system. The log files used in the research were automatically created by the environment from students' actions. Data in the log files consists of a) the username of the student in the game, b) the student's action (e.g. entering in programming activity, clicking on the help button, sending message), c) the precise time (date and time) the action was done. Data in the log files are used by the system to construct the student model at runtime.

Finally, the first written test (pre-test) was consisted of two programming activities about the research concepts. The goal of the pre-test was to extract students' prior knowledge. The second written test (post-test) was carried out with the completion of the research. It was consisted of five closed type questions to control further knowledge of the concepts and two programming activities. The programming activities were the same as in the first written test.

# 4 Results

## 4.1 Results from the 1st Empirical study

The results of the 1st study, according to the score of the questionnaires and the students' answers, can be divided in three categories: system usability, the usage of the system as an educational tool for teaching programming and the capability of designing and embedding new programming activities. The results for these categories were:

- **The usability of the system**: According to the score of Computer System Usability Questionnaire, the rating of the system usability is high. The final mean of the score in each question and for all the system components is 5.18 for a maximum of 7 (SD 1.2). Making the reduction in [0, 100] scale it arises a total score of 74/100 for the whole system usability. In spite of the high total scoring, there were two questions with a low score. These questions where question 9 that "system gives error messages that clearly tell me how to fix problems" where mean score was 3.78 (SD 1.8) and question 11: "the information (such as online help, on-screen messages, and other documentation) provided with system is clear" where mean score was 4.08 (SD 1.69). The low rating in these two questions shows that students need more guidance and better advises from the environment. In order to support these needs some improvements were made in online help and a system tutorial about the use and the capabilities of the mini-language programming tool was developed.
- **The usage of the system as an educational tool**: According to the university students' answers the system is pleasant, is attracting the interest, increases motivation, has a nice interface, is characterized by simplicity, functionality and ease of learning. Furthermore, one of its major advantages is that is web based and it is not depended on particular architecture or installations. Indicative students' comments

were: “Very amusing and satisfying”, “interesting activities with high level of creativity that it will attract curiosity and interest of students”, “Simplicity, ease of understanding and learning”, “with fantasy and desire you can make nice exercises for children”, “very nice interface, drag and drop f instructions is very useful”, “it is very positive that works in web browser”, “it is important that it does not need installation but only an internet connection, that simplify things”.
As a tool for supporting the teaching of programming the answers of university students in the research stated that it is very easy for someone to learn programming with the mini language, drag and drop of instructions makes programming easier, it eliminates syntax errors and is very useful for novice programmers. Indicative students’ comments were: “system simple use helps and encourage students to learn programming through gaming”, “very clever idea that differs from today traditional and boring method of teaching programming”, “an innovative method to introduce children in programming”, “drag and drop is a very intuitive method of inserting instructions and making a program eliminating syntax errors”, “tracing of the program in addition to execution visualization helps student to learn programming easier”.

- **Designing and embedding new programming activities**: In their work students designed several didactic scenarios that need the use of programming with system mini-language tool. Students designed prototype didactic scenarios that could be embedded in the knowledge base of the system. At the end they wrote down their thoughts and proposals about this task. According to students’ answers the design was very pleasant and developed their fantasy, instructions of mini language are sufficient for designing new didactic scenarios and proposed an authoring tool that could help in embodiment of the new activities in the game. Sample students’ answers were: “Scenarios design was very pleasant, I was feeling like designing a game”, “I was impressed! The design of didactic scenarios improved my fantasy limit”, “mini language has all the instructions I needed to design the scenario”, “an authoring tool could be helpful to try out the proposed scenario”.

### 4.2 Results from the 2nd Empirical study

The results of the 2nd empirical study are based on the scores of the questionnaires, log files and students’ grades in pre-test and post test and they can be divided in three categories: system usability, the utilization of the game as presented by the students’ models and the usefulness of the system as an educational tool for teaching programming to high school students. The results for these categories were:

- **The usability of the system**: According to the score of Computer System Usability Questionnaire, the rating of the system usability is high, like in the first research. The final mean of the score in each question for all the system components is 5.01 for a maximum of 7 (SD 1.6). Making the reduction in [0, 100] scale it arises a total score of 72/100 for the whole system usability. The interventions in the help component of the system made after the first research seemed to have positive effect on students’ evaluation of the system usability. More specific, question 9:

"system gives error messages that clearly tell me how to fix problems" the mean score was 4.85 (SD 1.3) showing an increase from the first research where mean was 3.78 (SD 1.8). Furthermore, on question 11: "the information (such as online help, on-screen messages, and other documentation) provided with system is clear" the mean score in the second research is 4.86 (SD 1.18) showing again an increase from 4.08 (SD 1.69). The demonstration of the game by the teacher played of course an important role in increasing these ratings. In spite of that increase the help component of the system seems to demand further development in order to provide more support to the students. From the students' answers in open type questions of the questionnaire was concluded that the game environment is pleasant, easy to use, arouses interest and supports motivation. From the observation of students' behavior it was clear that students were highly motivated and were trying to succeed in programming activities the same time they had fun. Sample comments were: "I'm satisfied with all", "it is comprehensible and funny", "It was an enjoyable and creative activity", "very easy handling, very nice in general" and "It could be better if there were more activities".

- **Students' models data and system utilization**: As already stated, the student model consists of data about the knowledge level of the student, the time spent in programming activities and also the time spent in other components of the system like help, chatting and wandering in the game world. The data for every action of a student are saved in log files automatically and permit the system to construct the individual learner model at runtime. From the log files of the second research the mean time of the overall system usage for every student in the experimental group calculated equal to 339.18 minutes. During this time students used the programming tool of the game for 235.16 minutes, searched for help for 40.7 minutes, chat with other students for 36.17 minutes and 17.13 minutes were spent on wandering in the game world.

  From the measures it is concluded that the 70% of the total time was spent on programming activities. The percentages make clear that the environment is doing well in arousing the interest and motivating the student to be engaged in programming activities. Furthermore from the total 235.16 minutes participating in programming activities, a percentage of 54% of the time was spent on selection control structures (if then else) and the rest 46% of the time was spent on repetition structures (while). Students of the experimental group worked on 7 programming activities, 4 for selection structure (57% of total activities) and 3 for the repetition one (43% of total activities). From this observation concludes a good analogy to the difficulty of the activities. Finally, as the level of knowledge is concerned, the mean of knowledge level of the students of the experimental group was70%. For the computation of the mean was used the mean of successful activities for every research concept, that was 74% in selection structure (2.94 successful to 4 total activities) and 75% in repetition structure (2 successful to 3 total activities). The result is in agreement with the mean grade of the experimental group in the post-test, which is presented below.
- **The usefulness of the system as an educational tool for teaching programming to high school students:** As stated above, during the 3rd week all of the students

participated in first written test (pre-test) and after that the students were randomly assigned to control group and experimental group. According to the statistical analysis from the t-test in pre-test grades there was no significant difference between the mean grades of the control group (mean=12.13, SD=8.27) and experimental group (mean=11.61, SD=8.11). On the other hand the grades in the second test (post-test) for the students in experimental group was significant higher (mean=68.61, SD=17.22) than those of the control group (mean=48.63, SD=21.18). The results of pre-test and post-test for the two groups are presented in table 1.

**Table 1.** Mean and SD for the two groups in pre-test and post-test

| **Students' Group - Method of Teaching – number of Students** | **Performance Pre-test Grade** | **Performance Post-test Grade** |
|---|---|---|
| Control Group - Traditional Teaching (n=32) | 12,13 (8,27) | 48,63 (21,18) |
| Experimental Group - Working with TALENT (n=33) | 11,61 (8,11) | 68,61 (17,22) |
| Total (n=65) | 11,87 (8,19) | 58.62 (19,2) |

From the results it is concluded that students in both groups, control and experimental, improved their performance by increasing the average rating from 11.87 (SD=8.19) in pre-test to average 58.62 (SD=19.2) in post-test. As implied by comparing the results of the two groups, students in experimental group had a significantly higher performance than students in control group. This indicates that there is some evidence that the use of TALENT contributes positively in learning programming concepts than the traditional method of teaching.

## 5 Conclusions

In the context of this thesis, we presented TALENT a web based adaptive educational multiplayer game for supporting the teaching of programming. TALENT is a combination of an adventure and a role play game and it utilizes web technologies in order to motivate the student to engage in programming activities. An innovative web programming tool which exploits visual programming techniques and embed an interpreter which executes student's program step by step. TALENT adapts to student abilities by providing adaptation in curriculum sequencing and adaptive navigation support. It also includes an authoring tool which can be used by an author to add or alter

educational content in knowledge base of the system or used by the teacher to monitor students' performance.

Two empirical studies have been conducted. The results from these studies are positive and indicate acceptance and satisfaction of participating students for the usability of the environment. Furthermore, the environment achieves to attract the interest of students and motivate them to keep on their programming efforts. Finally, the performance of students in an experimental group as opposed to these of a control group shows some evidence that teaching with TALENT has a positive contribution to the learning process in contrast to the traditional way of teaching.

Our future plans include the development of appropriate modules that would increase the collaboration between the students, the analysis of students' communication and also the exploitation of different adaptive methods and techniques in order to provide more personalized learning.

## References


1. T. Jenkins, "On the difficulty of learning to program", *In Proceedings of 3rd Annual LTSN_ICS Conference*, The Higher Education Academy, 2002, p.p. 53-58.
2. T.M. Connolly, and M.H. Stansfield, From eLearning to games-based eLearning: Using interactive technologies in teaching an IS course, *International Journal of Information Technology Management*, vol. 26, no. 4, 2007, p.p. 188-208.
3. T.M. Connolly, E. McLellan, M.H. Stansfield, J. Ramsay, and J. Sutherland, "Applying computer games concepts to teaching database analysis and design", *Proceedings of the International Conference on Computer Games, AI, Design and Education*, Reading, 2004.
4. M. R. Lepper, and T.W. Malone, "Intrinsic motivation and instructional effectiveness in computer-based education", In R. E. Snow & M. J. Farr (Eds.). *Aptitude, learning, and instruction: Vol. III. Cognitive and affective process analyses*, Hillsdale, NJ: Erlbaum, 1988, p.p.255-286.
5. S. Bennett, K. Maton, and L. Kervin, The digital natives debate: A critical review of the evidence, *British Journal of Educational Technology* vol.39, no.5, 2008, p.p.775–786.
6. P. Brusilovsky, E. Calabrese, J. Hvorecky, A. Kouchnirenko, and P. Miller, Mini-languages: A way to learn programming principles, *Education and Information Technologies*, vol. 2, no.1, 1997, p.p. 65-83.
7. R. Reichert, "Theory of computation as a vehicle for teaching fundamental concepts of computer science", Dissertation No. 15035, ETH Zürich, 2003.
8. M. Guzdial, and E. Soloway, Teaching the Nintendo generation to program, *Communications of the ACM*, vol.45, no.4, 2002, p.p.17–21.
9. C. Kelleher, and R. Pausch, (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers, A*CM Computing Surveys*, vol.37, no.2, 2005, p.p. 83-137.
10. J. Kay, "Learner Know Thyself: Student Models to Give Learner Control and Responsibility", *Proceedings of International Conference on Computers in Education, Association for the Advancement of Computing in Education (AACE)*, Z. Halim, T. Ottomann & Z. Razak (eds), 1997, p.p. 17-24.
11. K. Maragos, and M. Grigoriadou, "Towards the design of Intelligent Educational Gaming systems", *Proceedings of Workshop on Educational Games as Intelligent learning envi-*

*ronments, Artificial Intelligence in Education*, University of Amsterdam, Amsterdam, 2005.
12. S. Bull, "Supporting Learning with Open Learner Models", *4th Hellenic Conference with International Participation: Information and Communication Technologies in Education*, Athens, 2004.
13. Y. Cui, and S. Bull, Context and Learner Modelling for the Mobile. *Foreign Language Learner System*, vol.33, no.2, 2005, p.p. 353-367.
14. K. Maragos, and M. Grigoriadou, "Teaching Computer Science Concepts with Educational Computer Games", *Proceedings of EUTIC 2007 International Colloquium on Challenges and Uses of ICT Media and Information diffusion: towards an open society*, Laboratory of New Technologies in Communication, Education and the Mass Media of the University of Athens, Athens-Greece, 2007.
15. K. Maragos, and M. Grigoriadou, "Designing an Educational Onine Multiplayer Game for learning Programming", *Proceedings of Informatics Education Europe II*, ACM, Thessaloniki-Greece, 2007.
16. K. Maragos, and M. Grigoriadou, "Learner Modeling and Adapted Interaction in Educational Games", *Proceedings of 80Days' 1st International Open Workshop on Intelligent Personalization and Adaptation in Digital Educational Games*, University of Graz, Graz-Austria, 2009.
17. K. Maragos, and M. Grigoriadou, Exploiting Talent as a Tool for Teaching and Learning, *The International Journal of Learning*, vol.18, no.1, 2011, pp.431-440.
18. A. McFarlane, ed., *Information Technology and Authentic Learning: Realising the potential of computers in the primary classroom*. London: Routledge, 1997.
19. A. Mitchell, and C. Savill-Smith, The use of computer and video games for learning: A review of the literature, 2004; http://www.lsda.org.uk/files/PDF/1529.pdf [Retrieved 10/9/2005]
20. A. McFarlane, A. Sparrowhawk, and Y. Heald, Report on the Educational Use of Games, 2002; http://www.teem.org.uk/publications [ Retrieved 10/9/2005
21. R.M. Carro, A.M. Breda, G. Castillo, and A.L. Bajuelos, A Methodology for Developing Adaptive Educational-Game Environments, In P. De Bra, P. Brusilovsky, and R. Conejo(Eds.): *AH2002, LNCS 2347*, Springer- Verlag, Berlin Heidelber, 2002, p.p. 90-99..
22. A. Kerly, and S. Bull, Open Learner Models: Opinions of School Education Professionals, in K. Koedinger, R. Luckin & J. Greer (eds), *Artificial Intelligence in Education*, IOS Press, Amsterdam, 2007.
23. S. Bull, and M. McKay, "An Open Learner Model for Children and Teachers: Inspecting Knowledge Level of Individuals and Peers", *Intelligent Tutoring Systems: 7th International Conference*, Springer-Verlag, Berlin Heidelberg, 2004, p.p. 646-655.
24. J.R. Lewis, IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use, *International Journal of Human-Computer Interaction*, vol.7:, no.1, 1995, p.p. 57-78.
25. R. Likert, A Technique for the Measurement of Attitudes, *Archives of Psychology* vol.140, 1932, p.p. 1–55.