Using computer games as an aiding means in programming education

Mario Konecki, Bogdan Okreša Đurić, Luka Milić
University of Zagreb, Faculty of Organization and Informatics
mario.konecki@foi.hr, dokresa@foi.hr, luka.milic@foi.hr

Abstract

Programming is hard and demanding course that presents a problem for many computer science students. In order to try to make programming courses more interesting and engaging for students various approaches that use computer games as an aiding means in programming courses have been proposed and implemented. Computer games, as a well-known source of entertainment have been chosen because of their capability to capture the students' attention. In this paper a discussion about the problems in programming courses is given along with an overview of different approaches that include computer games in programming education. Discussion about the usage of these approaches is presented and conclusions based on presented work are given.

Keywords: programming education, problems, games, students
Main Conference Topic: Education, Teaching and E-learning

Introduction

Programming courses have always been a source of educational challenges for most of computer science studies and they are generally perceived and one of the more challenging courses (Baldwin & Kuljis, 2001; Gomes & Mendes, 2007). Problems connected to programming can be found on both students' and teachers' side. Some of the reasons for this kind of state lie in the fact that students are not accustomed to learn abstract matter. Their past education has usually consisted of mostly memorization and reproduction of facts, which is not sufficient when talking about programming. Programming has to be viewed as a skill because it involves implementation and usage of learned concepts and constructs, which should result in creative activities of developing various computer programs. Since failure and dropout rates in programming courses are high in many cases (Yadin et al., 2011) various efforts have been made to improve this situation. Many visualization based aiding tools for students of programming courses have been developed and have showed encouraging results (Sorva et al., 2013). Nevertheless, problems in programming courses remain present and persistent.

Reasons of problems and difficulties in programming courses have been researched and many different reasons have been stated. Gomes & Mendes (2007) give the following set of reasons for the situation in programming courses:

- Programming demands a high abstraction level
- Programming needs a good level of both knowledge and practical problem solving techniques
- Programming requires a very practical and intensive study, which is quite different from what is required in many other courses (more based in theoretical knowledge, implying extensive reading and some memorization)
- Usually teaching cannot be individualized, due to the common class size
- Programming is mostly dynamic, but usually taught using static materials
- Teachers' methodologies in many cases do not take into consideration students' preferred learning styles

- Programming languages have a very complex syntax with characteristics defined for professional use and not with pedagogical motivations

Another important aspect of programming education is the importance of motivation of students to go through their programming courses (Alaoutinen & Smolander, 2010). Students are frequently not motivated enough and have no clear idea about the purpose of their programming courses. Research has shown that no more than 22% of students have taken their programming courses because of the genuine interest, 40% of students have taken programming courses because of career reasons and 35% because it was obligatory to do so (Bergin & Reilly, 2005).

Because of many problems that are repeating in every academic year, a new approaches and continuing research is needed in order to find new means of increasing the results of programming courses. One of the possible approaches in programming education is the usage of computer games for educational purposes. In this way students should become more involved and more interested in their programming activities. By combining learning with entertainment medium, students are given a chance to learn in a more relaxed and less stressing environment. An overview of difficulties in programming education and an overview of efforts and research results in the area of using computer games in programming education are given in the rest of this paper.

Problems and difficulties in programming education

As is shown in studies mentioned further in this paper, motivation is a strong driving force for students and has a great impact on success of their studying. Motivation of students largely depends on their preferred learning style and individual preferences (Feldgen & Clua, 2004). Based on these diverse learning styles, four stages of learning programming concepts are recognized in (Li & Watson, 2011): receiving, visualizing, reinforcing, and applying and synthesizing. These stages are somewhat similar and match with categories of programming activities provided by Long (2007): discovering the algorithms, designing the architecture, writing the code, and testing and debugging.

Similarity of the mentioned stages and categories indicates that even though knowledge of code and instructions is necessary, students must become aware of the important aspects and concepts of programming. These aspects and concepts, which represent the basis for all programming activities of some acceptable quality, can be expressed through two activities: understanding of the problem and designing a solution that can be written using the known instruction set (Feldgen & Clua, 2004).

The most common students' and teachers' problems in the first course in programming around the UK were presented in (Milne & Rowe, 2002). Using an online questionnaire, the authors assumed that students, when providing answers to questionnaires, tend to give wrong answers, not because they are dishonest, but because they may believe they understand a topic, although the reality might be different. This concept of students tending to overestimate their understanding of a topic is not a novelty (Alaoutinen & Smolander, 2010). After ranking programming concepts in order of difficulty, according to the questionnaire answers, final analysis proved that pointers and memory topics cause the most problems. In order to counter these problems, authors propose that students should be provided with a thorough grounding in a procedural programming before they are taught object-oriented languages.

Authors in (Lahtinen et al., 2005) conducted a survey with the general goal of studying situation regarding Java and C++ courses, and finding out about the perceptions of students on some related aspects. When analyzing results of the survey conducted in several different European countries, some interesting conclusions were made: more than half surveyed students already had experience in programming, and most of them used programming language C++ for the purpose of learning the basics of programming. The most difficult

issues were of designing nature: understanding how to design a program to solve a certain task, dividing functionality into procedures and finding bugs in created programs (Lahtinen et al., 2005). Another experiment, described in (Feldgen & Clua, 2004) tries to solve exactly the problems stated above by using role-playing. The most difficult programming concepts in the mentioned study by Lahtinen et al. (2005) were: recursion, pointers and references, abstract data types, error handling, and using the language libraries. These results are somewhat similar to those presented in (Milne & Rowe, 2002), mentioned earlier. The problem seems to be in the students rather than in the teachers though. Students tend to understand the concepts, but have problems in learning how to apply them. Furthermore, students often overestimate their understanding of what they learned.

Another similar study was conducted at the high school level (Grandell et al., 2006) and its results are also mentioned in (Mannila, 2007). This study has shown that the most difficult concepts to be grasped for high school students are: algorithms, subroutines, exception handling and documentation. Mannilla (2007) reported on a study, based on the Structure of the Observed Learning Outcomes taxonomy (SOLO, a taxonomy that can be used to set learning objectives for particular stages of learning), which describes how code is understood by novice programmers using five SOLO levels. In accordance with mentioned study described in (Lahtinen et al., 2005), it seems that novice programmers can grasp the meaning of programming concepts in isolation, and that they mostly encounter difficulties when trying to explain what program does as a whole. Mentioned programming concepts and activities that have been reported as the most difficult from previously mentioned studies can be seen in Table 1.

Study A	**Study B**	**Study C**
pointer and memory topics	recursion, pointers and references, abstract data types, error handling, and using the language libraries	algorithms, subroutines, exception handling and documentation

Table 1: The most difficult programming concepts according to studies in (Milne & Rowe, 2002), (Lahtinen et al., 2005) and (Grandell et al., 2006) respectively

Supporting some of the mentioned studies, Ibrahim (2011) references a study which revealed that programming students fail because of incompetent lecturers, but because of incompetent students as well. Several supportive reasons are stated: did not understand lecturer's explanations, did not finish course tutorial, copying from friends, not even trying to answer questions from the main text book (Ibrahim, 2011). Since both students and teachers are possible problem starters, a new medium which would be more appropriate to students is a possible solution.

Ideal educational computer game, in general, is described by Ibrahim (2011) as a "game which combines both game design principles and learning theories with integration of learning content of specific subject or learning outcomes purposely to enrich learning for intended audiences". In his study, Ibrahim (2011) has measured perceptions of students towards using educational games for self-learning of introductory programming course by using five constructs: motivation, attitude, cognitive development, games interface and expectation. Success of games in education is visible in results of this study, which reports that a majority of the students think games make studying more interesting, and also that the majority of students prefer doing their exercises using games rather than doing the exercises in a traditional way during the class. In addition to helping them to think critically, students prefer self-paced studying offered by games.

Computer games in programming education

Although the idea of learning through games is not a new concept, movement for using computer games in education started in 2003 (Annetta, 2008). In his paper Jayakanthan (2002) proposes main types of benefits computer games can offer to their players, arguing towards usage of games in education: agency, immersion, challenge, reward, immediacy, variety, physical and mental engagement, and multi-sensory stimulation. Furthermore, several authors (Ibrahim, 2011; Li & Watson, 2011; Long, 2007) support the claims that instant gratification, one of the effects of playing a computer game, can speed up the learning cycle. According to Long (2007), instant gratification is only one part of three aspects every educational game should contain: simple start up since concentration and interest thresholds might be low, simple instructions to make learning of rules as simple as possible, and short modules which provide instant gratification.

Due to the stereotype that computer games contribute to obesity and antisocial behavior, implementing games into education on all levels might seem unwanted, but a research project of the United Kingdom has shown different results that speak in favor of games and their ability to encourage cooperation and critical thinking, thus developing independent social individuals (Kirriemuir, 2002). It has been found, as stated in (Annetta, 2008), that video games motivate by challenging and providing curiosity, beauty, fantasy, fun, and social recognition. Furthermore, games often generate positive emotional response in players, thus creating a more appealing and motivating learning experience. Critical thinking gained through gaming stimulates learning and gaining knowledge embedded in the game.

Interesting effect of games in education is mentioned in (Mayo, 2007), stating that using games in education significantly increased depth and complexity of what was learned, even though the breadth of the content learned did not change compared to no-game environment. Games in education are supported by Bond (2007) as well, for their promotion of exploration, experimentation, competition and co-operation as involvement techniques. Some aspects which help to build skills needed in the information age (e.g. science and mathematics skills, creativity, information and communication technologies skills, and the ability to solve complex problems) are also mentioned (Annetta, 2008).

Rarely met in papers on games in education, potential problem of such game usage is mental mode of gaming which is commonly associated with amusement, fun and relaxation (Bond, 2007), which is in opposition with mental mode of learning that is usually associated with concentration and reflection in higher education. Bond (2007) proposes a conceptual framework for serious scenario-based game design, which should be more appealing to higher education, with the main goal of reduction of the efforts put into designing, developing and exploiting a game.

Several games for educational purposes are mentioned by Annetta (2008), as some of the first examples of games used in education, which existed prior to year 2008: Immune Attack, Food Force, Discover Babylon and Quest Atlantis.

Jayakanthan (2002) mentions several earlier efforts undertaken to develop computer games to be used in education: Logo programming language as one of the first games ever that were developed primarily with educational purpose, E-GEMS as a project which has produced several important guidelines for educative games development, customized simulation games used in military training, and adventure games in business and corporate environments. Jayakanthan (2002) stresses three key areas which should play a central part in competency of an educative game developer: interface design (taking into account usability, accessibility, interactivity, etc.), domain expertise (primarily in order to make the game an intellectually stimulating experience for the player), and pedagogical knowledge (how knowledge should be presented to achieve the form that will reach the learner).

In her paper, Mayo (2007) emphasizes that a 6.542-student meta-study in the USA showed that students attending classes which offered some kind of interactive engagement reached higher score on the Force Concept Inventory. Interestingly, according to the cited study (Hake, 1998), the quality of lecture has no effect on the depth of understanding achieved by students.

Arguing the potential of video games in addressing systematic deficiencies of the USA science and engineering education system, Mayo, 2007 offers several supportive reasons: massive reach of games (numbered in 100.000's), effective learning paradigms (already in use in order to captivate their players), enhanced brain chemistry (stating that playing video games stimulates dopamine release, which in turn makes the brain ready for learning thus possibly increasing retention of game-presented educational material), time on task (stating that an average gamer in the USA spends about 6,8 hours a week on video games), and learning outcomes data (arguing that active engagement, as added value to visual representation, stimulates improved learning).

An interesting approach to using games in education is presented in (Li & Watson, 2011): a blend of programming learning tasks with the game construction process using tile-based games. In general, the authors recognize three different approaches to games in education: authoring-based approach (main learning activity is game development), play-based approach (programming is taught through playing a game), and visualization-based approach (presenting a mix of the former two).

In (Leutenegger & Edgington, 2007) the authors analyzed students' success on their computer science courses which involved programming. Layout of their course that has been presented consists of three quarters: first quarter consists of teaching Flash development environment and ActionScript syntax, second quarter consists of C++ programming language and Unix operating system, and the third quarter continues with C++ along with UML class diagrams and sequence diagrams which are also introduced (Leutenegger & Edgington, 2007). As argued through reasons why they chose languages as specified, a major asset of their approach is instant visual feedback for the students. In this way students have an opportunity to learn a language that has immediate applications, and elementary ActionScript is almost identical to C++ which is a major motivator for students since the game industry is heavily dependent on C and C++ (Leutenegger & Edgington, 2007). Students liked this approach and found it fun.

In her paper, Long (2007) focuses on IBM Robocode online programming game which demands players to play with a robot using Java. Players compete online in an arena-like environment where each robot tries to search and destroy other robots while protecting itself (Long, 2007). Everything a robot does is the result of a program provided by its player. Even though a large percentage of participants of one survey (Lepper & Malone, 1987) reported that their programming skills increased while playing Robocode, players of the game have shown various education levels. Interestingly, according to the survey of Long (2007), the most common reason for participating in the game was simply to have fun. Creativity and autonomy seem to be the factors which mostly made Robocode fun. Out of the four categories of programming activities the author mentioned, the most enjoyable one was discovering algorithms, closely followed by designing architecture, which means that the creativity is valued among students.

Feldgen and Clúa (2004) in their paper delve into motivation, learning styles, and role playing. After discovering that most freshmen have no idea why they have to learn programming, although they perceive it as a mandatory part of the chosen degree, and that they are not sure what a software engineer does (except of building Web sites and games), they came up with a role-playing game for their students. In this game, students play the role of designers and programmers. While one student writes specification and use cases of the

game, the other writes code for the specification. The programmer gives specification and the code to the designer, and they both have to verify the program. In the case that the program does not work as expected, designer has to rewrite specifications and give them back to the programmer. The authors found out that final result of the game largely depends on learning styles of the involved students - programs of mixed learning style students had a better structure than that of the same learning style students (Feldgen & Clua, 2004). Additionally, this sort of game develops students' social skills and their sense of achievement.

Games in education, and especially computer games, can come in several different genres. Role-playing game was mentioned earlier in this paper, and is described in (Feldgen & Clua, 2004), and a competitive-styled game is presented in (Lawrence, 2004). Both of these, competitive game and the Robocode, which is described in detail in (Long, 2007), are styled as online multi-player games, since they allow simultaneous interaction of several players using the Internet. Lawrence (2004) devised a competitive game which was open for participation of students during a project in their introductory data structures courses.

This interactive competition which was open during their assignment significantly increased students' efforts and satisfaction when compared to projects where competition part comes after the assignment is completed. In order to participate in the competition, students had to write a program which played a simple game. After writing the program, each student had to upload their work to the server, and choose to put their program to the test in a tournament with another program (made by a student or the lecturer). Such a workflow proved to increase students' motivation for solving their assignments and for their participation in the project.

Conclusions

Programming courses are important part of any computer science study. Many problems and difficulties, which are of recurring nature, have been recognized during programming education of a number of students in a number of academic years. Research has been done in order to find the reasons of persisting problems in programming education and the results have shown that there are many different reasons for this kind of state. Along with many reasons, many different approaches have been devised for improving the results of programming courses. One of these approaches is the usage of computer games to aid students in learning of programming concepts.

Computer games provide students with an entertaining medium that enables them to learn in a more relaxed and convenient way, which makes them more motivated to learn. In this paper an overview of different research efforts that involve using computer games in programming education have been presented and discussed. Based on presented research it can be concluded that using computer games in programming education has shown promising results although this kind of approach is not widely used in practice. More research in the area of programming education and using computer games as an aiding means is needed to devise a widely acceptable solution.

Brief biographies of the authors

Mario Konecki

Mario Konecki, PhD is a senior research and teaching assistant at the Faculty of Organization and Informatics in Varaždin, Croatia. During his former scientific work he has published over 40 scientific papers and has been actively involved in 8 scientific projects. He is also active in professional work in the field of programming, design and education. His main scientific interests are: intelligent systems, development of programming languages, education in the area of programming, design of user interfaces and web technologies.

Bogdan Okreša Đurić

Bogdan Okreša Đurić is a doctoral candidate of the Artificial Intelligence Laboratory at the Faculty of Organization and Informatics, employed on the project ModelMMORPG - Large-Scale Multi-Agent Modeling of Massively Multi-Player On-Line Role-Playing Games. His former education at the mentioned faculty, enriched by the international activity he experienced while studying, motivated him to pursue academic and scientific career in diverse fields related to semantic modeling, multiagent systems and social networks analysis.

Luka Milić

Luka Milić is a research and teaching assistant at the Faculty of Organization and Informatics in Croatia and a member of the Department of Computing and Technology. He graduated at the Faculty of Electrical Engineering and Computing in Zagreb, Croatia. His main research interests are: computer operating systems, internet of things and programming.

References

1. Alaoutinen, S, & Smolander, K. (2010). Student self-assessment in a programming course using bloom's revised taxonomy. In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (pp. 155-159). ACM, New York, NY, USA.
2. Annetta, L. A. (2008). Video Games in Education: Why They Should Be Used and How They Are Being Used. Theory Into Practice, 47(3), 229-239. doi:10.1080/00405840802153940.
3. Baldwin, L. P. & Kuljis, J. (2001). Learning programming using program visualization techniques. In Proceedings of the 34th Annual Hawaii International Conference on System Sciences (pp. 1051-1058). IEEE, Washington, DC, USA.
4. Bergin, S. & Reilly, R. (2005). The influence of motivation and comfort-level on learning to program. In Proceedings of the 17th Annual Workshop on the Psychology of Programming Interest Group (pp. 293-304). Psychology of Programming Interest Group, UK.
5. Bond, C. (2007). Serious Games for Higher Education: a Framework for Reducing Design Complexity. Journal of Computer Assisted Learning, 3(1), 1-23.
6. Feldgen, M., & Clua, O. (2004). Games as a motivation for freshman students to learn programming. In 34th Annual Frontiers in Education (pp. 1079-1084). IEEE. doi:10.1109/FIE.2004.1408712.
7. Gomes, A. & Mendes, A. J. (2007). An environment to improve programming education. In Proceedings of the 2007 International Conference on Computer Systems and Technologies, Art. No. 88 (pp. 1-6). Rousse, Bulgaria, ACM, New York, USA.
8. Grandell, L., Peltomäki, M., Back, R.-J., & Salakoski, T. (2006). Why complicate things?: introducing programming in high school using Python, pp. 71-80.
9. Hake, R. R. (1998). Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. American Journal of Physics, 66(1), 64-74. doi:10.1119/1.18809.
10. Ibrahim, R. (2011). Students Perceptions of Using Educational Games to Learn Introductory Programming. Computer and Information Science, 4(1), 205-216. doi:10.5539/cis.v4n1p205.
11. Jayakanthan, R. (2002). Application of computer games in the field of education. The Electronic Library. doi:10.1108/02640470210697471.
12. Kirriemuir, J. (2002). Video Gaming, Education and Digital Learning Technologies. D-Lib Magazine, 8(2), 25-32. doi:10.1045/february2002-kirriemuir.

13. Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. ACM SIGCSE Bulletin, 37(3), 14-18. doi:10.1145/1151954.1067453.
14. Lawrence, R. (2004). Teaching data structures using competitive games. IEEE Transactions on Education, 47(4), 459-466. doi:10.1109/TE.2004.825053.
15. Lepper, M. R., & Malone, T. W. (1987). Intrinsic motivation and instructional effectiveness in computer-based education. In Aptitude, Learning, and Instruction (pp. 255-286). Hillsdale, NJ: Erlbaum.
16. Leutenegger, S., & Edgington, J. (2007). A games first approach to teaching introductory programming. In Proceedings of the 38th SIGCSE technical symposium on Computer science education - SIGCSE '07 (pp. 115-118). New York, USA: ACM Press. doi:10.1145/1227310.1227352.
17. Li, F. W. B., & Watson, C. (2011). Game-Based Concept Visualization for Learning Programming. In Proceedings of the third international ACM workshop on Multimedia technologies for distance learning (pp. 37-42). ACM.
18. Long, J. (2007). Just For Fun: Using Programming Games in Software Programming Training and Education. Journal of Information Technology Education: Research, 6(1), 279-290.
19. Mannila, L. (2007). Novices' progress in introductory programming courses. Informatics in Education, 6(1), 139-152.
20. Mayo, M. J. (2007). Games for Science and Engineering Education. Communications of the ACM, 50(7), 30-35.
21. Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming-views of students and tutors. Education and Information Technologies, 7(1), 55-66. doi:10.1023/A:1015362608943.
22. Sorva, J.; Karavirta, V. & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. ACM Transactions on Computing Education (TOCE), 13(4), 15.
23. Yadin, A. (2011). Reducing the dropout rate in an introductory programming course. ACM Inroads, 2(4), 71-76.