This is the published version of a paper presented at European Conference on Games Based Learning 2017.

Citation for the original published paper:

Shabalina, O., Malliarakis, C., Tomos, F., Mozelius, P. (2017)
Game-Based Learning for Learning to Program: From Learning Through Play to Learning Through Game Development.
In: Maja Pivec & Josef Gründler (ed.), 11th European Conference on Games Based Learning (pp. 571-576). Reading, UK: Academic Conferences and Publishing International Limited
Proceedings of the European Conference on Games Based Learning

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:miun:diva-31797
Abstract: This research is approaching game-based learning as a creative act. Furthermore, creativity receives a holistic interpretation and is regarded as a three-dimensional term. Consequently, the authors will debate the research topic of game-based learning from three perspectives: teacher, student and learning process. On the other hand, computer programming is still hard to learn; as many researchers show that students fail vastly in programming courses, game-based learning seems to be a motivational solution that overcomes the basic difficulties that students face in this field. In the emerging field of game-based learning, there are several approaches that have been tested in programming courses at university level. Three of them are a) Learning to program by playing games, b) Learning programming through game development and c) Learning to program by the development of games for learning programming. This study has been carried out as a comparative analysis of these three approaches. Data has been collected from course evaluation questionnaires, online discussion fora and from discussions with students and teachers involved in programming courses at three different universities. Findings show that all three approaches seem to have a promising potential but there are also some identified minor problems. The recommendation is to blend these approaches and to combine them with traditional instructional design for programming education. The outcome of the study is the design of a model of learning for students in programming education.

Keywords: programming education, game-based learning game construction, creativity, model of learning, blended learning

1. Introduction

Computer programming at university level have in several studies been identified as a problematic area with low pass rates and poor learning outcomes (Guzdial & Soloway 2002; Lahtinen, Ala-Mutka & Järvinen, 2005; Mozelius & Olsson, 2015). Classical learning technologies are becoming less effective than before. Modern students find today’s education boring and complex. At the same time, the new generation of students today play more digital games than ever before (Juul, 2010). Research studies have reported about the learning potential of using games in educational contexts (Malone & Lepper 1987; Gee 2003; Prensky 2007), and there are several reasons for testing the potential of Game-based learning (GBL) in programming education. Computer programming is one of the few subjects where students can use computer games not only as learning systems but build also their own games. And besides, playing games and games development tend to train creative thinking that is on the most challenging problems in programming education.

2. Game-based learning as a creative act

Programming courses in Higher Education Institutions encounter many challenges, as students find difficult to learn to program. Moreover, researchers also discovered positive and negative results from using technology (Rashid and Asghar, 2016). However, it is also a challenge, for both educators and researchers, in meeting students' requirements and finding the best way to communicate the knowledge, skills and capabilities students need in order to be able to understand programming.

Therefore, it is crucial to have a three dimensional perspective when approaching programming courses: students, teacher and the learning process. Moreover, it is advisable to see programming education and the approaches to learning embedded within it, as a creative act. Consequently, the authors of this study considered that is essential to explain the meaning and the role of ‘creativity’ and provide a summary of the research literature that debated this topic.
This study supports and develops on Shabalina et al. (2016). Shabalina et al. (2016) described creativity from three perspectives: teacher’s perspective of creativity and the creative pedagogy, students’ creativity and the process of creativity itself, including learning. Nevertheless, Ayob et al. (2011) depicted creativity by means of experiential learning, in other words, as a process mainly, the process of creative learning. Further, they suggested that creativity can be developed through ‘hands-on’. It follows that, experiential learning has the capability to nurture creativity and innovation and not only, but also, to lead towards student skills development, such as problem solving (Ayob et al., 2011). This assertion is rooted in Kolb’s (1984) research on experiential learning. Kolb (1984) described experiential learning, which is crucial for creativity, as a process of gradual learning by building successively more and more knowledge. This argument is supported by Feinstein (2011) in the research on learning patterns for creativity in a field. In addition to the process of creative learning perspective, Ayob et al. (2011) argued about a trait perspective, where students acquire a wide range of abilities through experiential learning. These could be named ‘creative skills/abilities’ which are required when learning programming for instance. We, thus, enumerate just a few out of the eleven discovered creative capabilities, which were gathered by students within the experiential learning: ability to produce alternatives, problem awareness, ability to develop or present further detail, ability to highlight the essence, internal visualization, humour, flexibility and combination and synthesis. The definition of ‘creative traits’ consists of the combination of all these capabilities required within the learning process (Ayob et al., 2011). The findings are validated by using a range of measurements and tests (Torrance and Safter, 1999) and mixed methods research (observation, creativity tests, survey, portfolios, focus groups).

Further research on creativity was conducted by Bertoncelli, Mayer and Lynass (2016). They argued that creativity was an essential element within both research world and engineering. As a consequence, it is suggested that could be also an aspect of programming. The authors of the study linked together creativity and learning and emphasised the fact that a profound understanding of learning, would lead to an increase of creativity. It was also highlighted that creativity is no more an individual characteristics of a certain individual or domain, but instead, it belongs to large international groups of people or communities, and embeds interdisciplinary aspects within the ‘learning environments’ (Bertoncelli, Mayer and Lynass, 2016; Sawyer, 2012; Tomos et al., 2016). In addition to being encouraged by universities and companies, creativity was recommended as a discipline and Bertoncelli, Mayer and Lynass (2016) mentioned in their study about the training for creativity and specific programs designed to encourage creativity skills in solving problems. They demonstrated in their study the link between learning and creativity training. Finally, Romero, Hyvönen and Barberá (2012) stated that creativity is a human capability that is acquired along the human life.

By contrast to the above selected research literature, this study will approach programming through game-based learning and will highlight and define creativity based on three perspectives: students’ perspective, teacher’s perspective and the creative learning process itself. The study will construct a model of learning in programming education. The novelty of this approach consists of combining the three perspectives as well as the three approaches to learning programming conducted across three European countries, at three universities (Stockholm/Sweden, Volgograd/Russia and Thessaloniki/Greece): (a) learning to program by playing games; (b) learning programming through game development; (c) learning to program by the development of games for learning programming.

3. Approaches to game-based learning and related theories

There is lack of research on creativity and approaches to learning programming through game-based learning (Shabalina et al., 2016). This study will build on previous research and will fill the gap in knowledge by constructing a Conceptual Model of Learning Programming through Game-based learning. Previous research indicated a relationship between student engagement and ownership, teacher as a facilitator, together with the embedment of pedagogical principles and the creative learning process (Shabalina et al., 2016: Figure 1: A Conceptual Model of Creativity). It follows that game-based approach is a creative technique, aiming to facilitate learning programming. Although, game-based learning is a student-centred learning technique (Yang and Chang, 2013), the literature indicates the existence of three types of digital games: commercial games, professional games developed by educators and student-designed games (Ebner and Holzinger, 2007). This study refers to (1) educational games developed by instructors for students, which are related to the learning approach: (a) learning programming by playing games and (2) student-designed games, which refers to the second approach to learning programming, named (b) learning programming through game development. However, what it is novel and indeed differentiates this study from other research, it is the third approach to
learning programming, which requires (c) learning to program by the development of games for learning programming. In other words, (3) the students build serious games about the path and stages of learning the real process of programming. In this manner, they have an initiation in the programming by 'hands-on', by exploring in a creative manner and developing in depth thinking which leads to understanding and finally to the deep learning of programming. This is in fact the digital game authorship (DGA) mentioned by Yang and Chang (2013) and which is also the result of social constructivist learning theory, and the individualization of learning with the collaboration and help of the teacher. It requires participation, alternatives, development, problem solving, creativity, combination and entertainment/humour (Ayob et al., 2011). In other words, this is the approach to learning programming through game-based learning and design and involves high intellectual skills and confirms research by Prensky (2007), the experiential learning theory (Kolb, 1984) and the game theory (Binmore, 1994; Burguillo, 2010).

4. Game-Based approaches in learning programming

4.1 Learning programming through playing games

Several research studies have been carried out and corresponding games have been developed that aim to include the aforementioned features to support learning and teaching focusing on the computer programming domain. The majority of these games include a specific scenario that aims to cover a specific computer programming unit, while fewer games cover multiple learning objectives and theory units.

CMX was designed and developed to enhance learning and teaching of computer programming and to be used in the classroom as a supporting tool (Malliarakis et al., 2013a; Malliarakis et al., 2014b). It aims to increase students’ participation so that they are able to practice more with the concepts they are taught, without however replacing the teacher’s role as the tutor.

The main environment of CMX replicates a toxic factory which pollutes the ecosystem with toxic waste, putting in danger the last remaining land of the world. In this alternate reality, a team of individuals called crackers are activists that are trying to invade the factory and shut down its main server so that it stops polluting the environment. However, the factory is equipped with employees named hackers, who are paid to protect the server and the factory’s on-going operation. A virus has infected the main server, and has made the server vulnerable to attacks. Thus, the crackers are seizing this opportunity to find the passwords hidden inside the factory and reach the server to enter them and shut it down, while hackers are trying to find the passwords in order to destroy the virus (Malliarakis et al., 2013c).

Students are required to participate in two phases where they are taught the programming language C. It should be noted that the game can be easily customized to support other programming languages such as Java etc. The first phase represents a training process, where each player of both teams is trained individually by specialized characters existing inside the world, called Senseis. This training process is essential, as students can learn the theory regarding programming concepts and can also practice on the taught materials with the help of the Senseis. This way, they can accomplish their missions and proceed to the next levels, where each level provides them with one of the required passwords and brings them closer to the factory’s main server. More specifically, there are three different levels that the students need to pass, each represented by a different type of Sensei (Senseis, Iron Senseis and Golden Senseis). Each Sensei can provide students with a password that unlocks a specific Sensei of the next level, if they execute the requested tasks successfully. The students initially try and locate the first level of Senseis within the graphical environment. Then, students perform an interactive dialogue with each Sensei, where they are required to respond to multiple-choice and right/wrong questions in order to receive the secret passwords (Malliarakis et al., 2014a; Malliarakis et al., 2014c).

Once the first set of passwords has been located, students proceed to find the Iron Senseis. Then, they enter the passwords so that the Iron Senseis can identify that each student has already passed the first level of the training. Once the authentication process is complete, students are required to study a series of tiles of codes that together form an executable C program, and then construct that code by dragging & dropping and placing the tiles in the correct order. The system examines the code submitted and provides corresponding feedback to each student. If the code is wrong, the system presents an error message and allows the student to try again. If the code is correct, the student is given a new set of passwords which he/she has to provide to the Golden Senseis for authentication.
During the last training phase, students are required to write actual programs using commands of the C language on a programming editor provided within their interactions with the Golden Senseis. The code is then compiled by an embedded compiler and the result is either a message that the code is correct or error and explanatory messages when students have made mistakes so that they can fix them. Correct codes give the students the final password that provides access to the main server (Malliarakis et al., 2016).

4.2 Learning programming through game development

The analysed course is an introductory programming course at university with the dual aim to teach Multimedia programming and try to attract new groups of students to programmes at a department of computer and systems sciences. The course syllabus was built around the idea that multimedia and game construction can be a stimulating way to learn useful programming techniques. Almost all course content is created and provided in an overloaded multimodal model in an online environment. Lectures, tutorials and the course book are all built around analysis and synthesis of digital games and provided in a classroom that was flipped before the term was coined. The course developers’ idea has been to give students access to all course content as early as possible and during the rest of their lives. Unfortunately this has later been restricted by the departments’ general restrictions.

All assignments and the final mini-project have been built around the idea of learning to program by game construction. In the final project students should implement their own designed digital game, provided with a tutorial and a demonstration video explaining the basic gameplay. Submissions to the assignment should also include a documentation of the design and development process. During the first course batches students had a free choice of building just any game, if the game idea met the grading criteria. Later students were required to construct an educational games only following more specified design guidelines. The main reason for restricting the free choice was to prevent plagiarism.

The mix of game construction and multimedia programming has worked fine for almost all learners in what must be classified as heterogeneous student groups. Course batches have had an age mix from 18 to 65 year old participants and with a wide geographically spread in a course where all course sections could be completed as pure online studies. Course outcomes have been better for students above 30 years old, but some younger digital natives that have started out early with programming have constructed games with excellent games and also contributed to the online discussions.

Instructional design has been created with influences from several pedagogical models but with constructionism as the main learning theory. Constructionism is based upon the concept of constructivism and shares the belief that learning should be a process of “building knowledge structures through progressive internalisation of actions” but with the extension of “this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it’s a sand castle on the beach or a theory of the universe” (Papert, 1980). Seymour Papert had the idea of creating a ‘Mathland’ for students learning Mathematics, this course has had the idea of creating a ‘Gameland’ for learning to program by game construction.

4.3 Learning programming by the development of games for learning programming

Using games both as learning systems and design objects can be promising and attractive for teachers and students. Games as learning tools motivate students to learn and provide training skills. But development of such games requires qualified software engineers, experienced in game design, and also specialists in the field of education that could prepare learning materials (Fig.1).

![Figure 1: Learning programming through playing games](image-url)

Using computer games as design objects in teaching the program development process has also several important advantages. The purpose of games development is naturally understood by the majority of the
students; many students are familiar with this field and can form adequate requirements for this kind of programming systems. They are motivated to work on a game as they are interested to see the result. Game development also trains team work skills, the setting of project management priorities, planning and conflict resolution.

**Figure 2:** Learning programming through game development

But even more challenging is a combination of the two approaches (Shabalina et al, 2013), if students develop not just regular games but games for learning programming (Fig.3). In this case they have to cooperate with teachers familiar with programming languages in order to create learning content and to find a way of its integration to a game context. They have also to test their games on other students studying programming languages. In such a way the students study all the phases of the program development process. As a result of their work the teachers get learning games that can be used for teaching the next generation of students.

**Figure 3:** Learning programming through the development of games for learning programming

The approach has been implemented at Computer-Aided Department of Volgograd State Technical University for teaching programming. Every year several teams of students under the supervision of teachers develop games for learning programming languages and technologies. The games developed by the students are used for teaching programming and testing new students.

Students that participated in experiment and developed educational games expressed their opinion about the project. They consider game development as a very inspiring thing to do. Development of educational games helped them a lot in practicing skills they learned during studying. Also most of students noticed that they were motivated to learn many new things that are not included in the standard curriculum.

5. Conclusions

In the emerging field of game-based learning, there are several approaches that have been tested in programming courses at university level. Three of them are a) Learning to program by playing games, b) Learning programming through game development and c) Learning to program by the development of games for learning programming. This study has been carried out as a comparative analysis of these three approaches. Data has been collected from course evaluation questionnaires, online discussion fora and from discussions with students and teachers involved in programming courses at three different universities.

The recommendation is to blend the described approaches and to combine them with traditional instructional design for programming education. What seems like an interesting idea is to give game construction assignments to last year students to build learning games that can be played by students in introductory programming courses. The main problems could be the difficulty of organization of a team work and lack of time for coordinating them. But speaking generally this approach looks to be effective in using GBL for teaching programming.

Furthermore, findings confirm authors’ earlier findings that GBL is a promising concept to stimulate creativity and flow in programming courses (Shabalina et al., 2016). The recommendation is larger group projects where students work in teams, in the same way as games are developed in commercial companies.
References


