# CodeAdventure: An Adventure Game for Computer Science Education

Panayiotis Andreou[1(✉)], George Nicou[1], Irene Polycarpou[1], Panagiotis Germanakos[2], and Nearchos Paspallis[1]

[1] University of Central Lancashire, Larnaka, Cyprus
{pgandreou,gnicou,ipolycarpou,npaspallis}@uclan.ac.uk
[2] SAP SE, Walldorf, Germany
panagiotis.germanakos@sap.com

**Abstract.** Among the most fundamental concepts of computer science education is introductory programming, and, many effective approaches were proposed to teach programming to novice learners. Despite all efforts, students still show poor performance with course assessments, which can then lead to dropping out of their studies after their first experience with programming. In the last few years, a number of interventions were suggested, including the utilization of educational games that can motivate, stimulate and engage students far better than conventional approaches. In this paper, we present CodeAdventure, an educational adventure-like game that has been designed for learning and practicing introductory programming concepts. CodeAdventure adopts an integrated design approach that employs various mechanisms and techniques to achieve a truly immersive game learning experience while in parallel provides a fun way to practice and apply various programming concepts. CodeAdventure uses compelling graphics and incorporates different learning techniques that have been shown to be effective for students' learning, such as providing hints and clues on how to solve puzzles, referencing instructional material, and immediate feedback on students' performance .

**Keywords:** Game-based learning · Educational technologies · Learning · User experience · Programming concepts

## 1 Introduction

Among the most fundamental concepts of computer science education is introductory programming, and thus, many different approaches were proposed on efficient and effective ways to teach programming to novice learners. Despite all the efforts, it has been well documented that students have difficulties learning introductory programming and as a result, many of them have poor performance with course assessments, which can then lead to dropping out of computer science studies after their first experience with programming [2, 12]. In [2], the authors present the results of study which suggest that the main sources of student-reported learning difficulties are the lack of previous knowledge, the lack of effort or personal persistence, and the lack of motivation.


P. Zaphiris and A. Ioannou (Eds.): LCT 2017, Part I, LNCS 10295, pp. 423–432, 2017.
DOI: 10.1007/978-3-319-58509-3_33

In the last few years, a number of interventions have been suggested [4, 15–17], including the utilization of educational software [1, 9–11], and more specifically, educational games [3, 6, 13] that can provide students with a non-traditional learning environment in which they can learn and practice in a fun and intuitive way. Many researchers argue for the appeal of educational games to students of all ages and their potential to enhance the educational experience, including motivating, exciting and engaging the students in the learning process [5, 7, 8, 14]. Educational games can enable self-paced interactive learning; allow students to make mistakes and re-attempt problems without negative outcomes; support various modes of level difficulty and achievements to accommodate for mixed ability classes and personal student goals; and enhance the development of understanding through immersive environments and their reflective nature [3]. Furthermore, educational games can take advantage of students' enthusiasm about video games and serve as an additional motivational and inspirational factor. Given the demanding curriculum and the limited amount of time available for instructors to convey the rather complex (for novice learners) concepts of programming, an educational game can provide an additional resource of information and practice problems that students can use on their own time and pace as well as provide the opportunity to spend additional time to focus on the material that is more challenging for them.

In this paper, we present an educational game, called CodeAdventure, which has been designed for computer science students to learn and practice introductory programming concepts. CodeAdventure is an adventure game where the player assumes the role of the protagonist exploring the game environment, and in the process, solving puzzles, and overcoming challenges in order to discover and acquire certain items. In terms of educational value, in addition to providing a fun way to practice and apply various programming concepts, CodeAdventure incorporates different learning techniques that have been shown to be effective for students' learning, such as providing hints and clues on how to solve puzzles referencing instructional material, and immediate feedback on students' performance [6].

The remainder of this paper is organized as follows: Sect. 2 reviews the literature. Next, Sect. 3 presents the CodeAdventure educational game and finally Sect. 4 concludes the paper.

## 2 Background and Related Work

In the last few decades there has been a reform movement in higher education to shift from the traditional paradigm, the positivist approach, towards the constructivism paradigm. This emerged in a wide range of academic areas such as philosophy, the arts, education, politics, religion, medicine, physics, chemistry, ecology, evolution, psychology, linguistics, and sciences [25, 26]. Following the constructivist approach, educators serve as the main teaching instrument. According to [27], constructivist knowledge "is knowledge that human reason derives from experience. It does not represent a picture of the 'real' world but provides structure and organization to experience" (p. 5). For the educators the facilitation of understanding is a process of co-construction of multiple meanings in which they accommodate their own understanding to fit

students' own experiences. To be able to construct their own understanding, students should be actively involved and engaged in the learning process, rather than being passive recipients of knowledge. This turns to be one of the most challenging principles of constructivism and the one that educators struggle to accommodate.

Educators from different disciplines are investigating ways to enhance the way they design and develop their teaching material and activities, so that they are more motivational and engaging for the students. According to [20], failure of students to actively engage with the taught material can lead to poor understanding and performance with the course (e.g., low marks on the course assessments). Students learn best when they feel that they enjoy learning and are engaging in the learning process. Furthermore, the results of a study conducted by [22] suggest that the higher the student engagement, the higher the student learning. There are many suggestions in the literature on how to motivate and engage students in their own learning, especially as they relate to sciences. As reported in [20] students can be motivated when the material is presented within a real life context, in relation to students' needs, as well as when they are provided with constructive feedback, valuable reward for their efforts, and clear expectations.

One way to engage students is to provide an enjoyable learning environment through play and fun activities, a concept that many times is referred to as edutainment (entertainment which is specifically designed to be educational, i.e., educational entertainment). As with entertainment, edutainment can have many different forms, ranging from media (e.g., video productions, audio productions, computer software, etc.) to physical places (museums, educational centres, parks and exhibits, etc.), all with the aim to attract learners and keep their interest on specific subject areas. Of special interest to this paper, are educational software, and specifically, educational video games. Many researchers argue that educational video games, as opposed to other edutainment elements, have a greater appeal to students of all ages and can potentially enhance the educational experience. While edutainment is being grounded on only didactical purposes and allows for static linear progressions, [29], it does not allow for exploration or consideration of alternative routes, thus decreasing the overall motivation and engagement of the student. In contrast, the immersive environments supported by educational video games use compelling graphics, sound, and physical interaction that can significantly enhance the educational experience, motivate, excite and engage the students [5, 7, 8, 14, 18, 19, 24]. Additionally, allowing users to roam freely large spaces and explore different scenarios, which they can connect to their learning experience, increases the cognitive curiosity of the student and offers an intrinsic reward.

On the other hand, the majority of games also employ a number of extrinsic rewards for their end-users to further increase motivation, participation and engagement. This includes game components like points, levels, badges and quests that can be used also for tracking the players' progress. Allowing the user to reflect on his/her progress is also a very important element of active learning. In particular, educational games use concrete goals to make learning through reflection possible. In particular, by placing specific goals that the user must meet (e.g., quizzes and puzzles that assess the level of knowledge) the game ensures that essential skills are acquired before he/she can continue in more complex areas.

Educational games have gained increasing popularity in the last years and there has been a rapid growth in their development and effective use as non-traditional learning environments for independent learning as well as a supplement of traditional instruction, along with other pedagogical methods [5, 19]. They have become part of the modern culture and the everyday life of the young generation of learners, since they become familiar with them as early as preschool when they use them to learn numbers, letters, colours, shapes, etc. [5]. Computer Science and more specifically, introductory programming, is no exception. There is a vast literature on educational games and how they can be used to enhance instruction of introductory programming as well as to motivate and engage young learners with introductory programming [21, 23].

## 3 Code Adventure

CodeAdventure is an adventure game where the player assumes the role of the protagonist exploring the game environment, and in the process, solving puzzles, and overcoming challenges in order to discover and acquire certain items. CodeAdventure adopts an integrated design approach by employing various mechanisms and techniques to achieve a truly immersive game learning experience. In particular, the game has an attractive storyline and offers a highly interactive 3D environment that allows the user to explore different levels, which are composed of multiple rooms, each one containing a diverse set of puzzles. Each room has specific objectives and includes various encyclopaedic interactive elements that convey the required knowledge to meet the objectives as well assessment methods that allow the user to reflect on his/her current status and progress. Moreover, achievements, rewards and secret items provide a feeling of challenge and increase overall engagement.

In the following sections we thoroughly describe the design aspects of CodeAdventure.

### 3.1 Storyline

The game story takes place in the year 2500, where the player, who is a time-traveller, detects a strong ripple in the fabric of space-time caused by another time-traveller. This nemesis, who travelled back in time to steal the six rotors of the Enigma machine, has a mission to prevent Alan Turing from breaking Enigma ciphers, thus allowing the Nazi to win World War II. The player traces that the nemesis has travelled back in 1939, and hid the rotors in different rooms. The game starts at the point in time where the player has travelled back in time, and is searching to find the six rotors. The player is required to solve specific programming puzzles and overcome specific challenges (e.g., unlocking doors and chests) in order to explore all the rooms and discover the hidden rotors.

### 3.2 Level Design

Code Adventure features 4 distinct levels representing different “subject areas” for introductory programming. In particular, there are four levels incorporated in the current version of the game: (i) Introduction to the JAVA language; (ii) Object-oriented

Programming; (iii) Data Structures and Algorithms; and (iv) Advanced Topics. As illustrated in Fig. 1(left), each level is composed of multiple rooms that may have one or more doors[1] each posing unique challenges (e.g., quizzes, puzzles) that must be met in order to allow to the user to continue to next one. Each room has a specific theme that represents one or more related programming topics (e.g., introduction to data types) as can be seen from the superimposed overlay in the middle of Fig. 1(right).


**Fig. 1.** (left) One of the CodeAdventure levels showing the configuration of multiple rooms; (right) One of the rooms of CodeAdventure that introduces data types

### 3.3 Room Design

Each room of CodeAdventure focuses on one or more related introductory programming topics. At the entrance of the room, the user is informed about the learning objective(s) using an overlaid message, as illustrated in Fig. 1(right). When the user enters the room, he/she need to explore and find the entrance to the next room by finding specific coloured key cards. In order to acquire these cards, the user must explore the room, discover their location and solve specific challenges to get them.

Examples of challenges are illustrated in Fig. 2, where the user is asked to: (i) (top-left) solve a multiple choice question; (top-right) rotate a wheel to solve a puzzle; (iii) (bottom-left) rotate boxes with operators to validate a mathematical expression; and (iv) (bottom-right) pull/push levers to form the bit representation of a number.

In order to solve the challenges the user can participate in a number of learning of active and passive learning activities. These activities currently include: (i) static overlaid visual elements (e.g., banners in corridors, wall papers); (ii) interactive visual elements (e.g., movable boxes, push buttons, levers); (iii) animated elements (e.g., moving rotors, small toy vehicles); and (iv) reference points (e.g., question mark icons that link to external documentation). In order to facilitate our description, consider the gameplay screenshots in Fig. 3. As was mentioned before, each room has a specific theme that represents one or more related programming topic (e.g., introduction to data types, casting, operator precedence). As can be seen in Fig. 3(left) and (right) has an overlaid label that shows the type of each box. For example, Additionally,

[1] Doors include traditional doors, automatic doors, electric fences, etc.

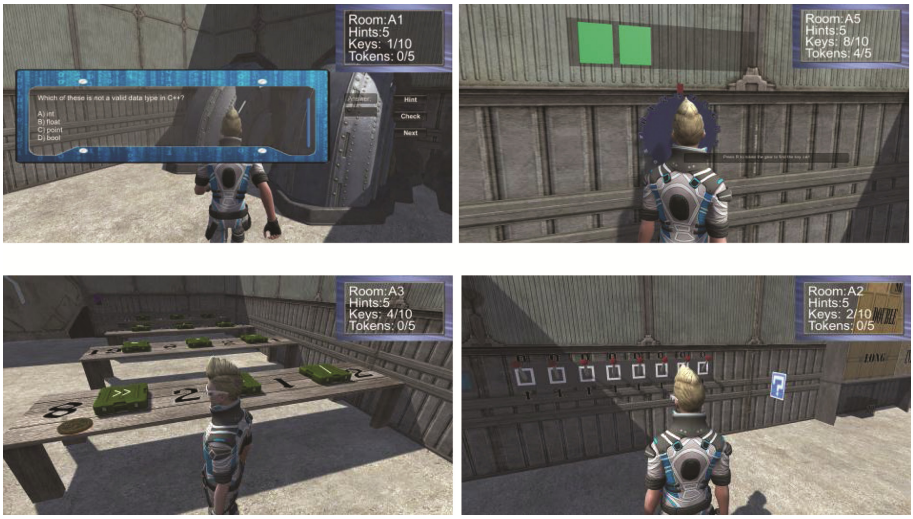


**Fig. 2.** Examples of challenges in each CodeAdventure room: (top-left) multiple choice question; (top-right) mechanical wheel puzzle; (bottom-left) rotating operator boxes mathematical challenge; and (bottom-right) pulling/pushing levels for bit representation of numbers


**Fig. 3.** (left) static visual elements: (a) boxes with overlaid data type name. (b) Quantified visual representation of data types: a single box represents a byte-sized data type such as byte and boolean. Boxes are grouped to form larger data types (e.g., two boxes form a two-byte data type such as short, four boxes form a four-byte data type such as int); (right) expressions embossed on corridor tiles.

### 3.4 End-User Monitoring

The monitoring of the end user interaction is being performed in a multi-modal way as illustrated in Fig. 4. All the actions of the user are recorded and time stamped for further analysis and understanding of how much time a student spent on a specific activity learning or practicing. This sets the foundation on understanding the user's behaviour and assessing the effectiveness of the software.

Additionally, we have also incorporated components to collect continuous streams of sensor data (e.g., Accelerometer, Galvanic Skin Response and Heart Rate) through smart wearable devices and to expose them in a unified way for further processing. These

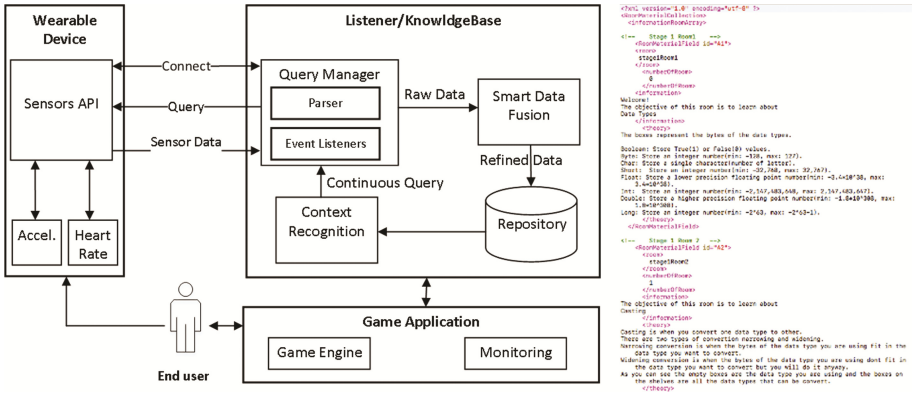


**Fig. 4.** (left) The monitoring architecture of CodeAdventure (right) XML representation of the information utilized for room configuration

mechanisms will in the future augment the data received from the aforementioned monitoring of activities to infer knowledge about a user's affective state during game play, eventually leading to an enhanced understanding of his behaviour and experience.

### 3.5 Feedback

Allowing the user to reflect on his/her progress is a very important element of active learning. To this end, CodeAdventure provides a variety of feedback mechanisms that allow the student to be informed on his progress and assess his current level of knowledge.

On the top right corner, the user is always aware of his location (i.e., level and room code), number of available hints, and number of keys and tokens acquired. Before entering the room, an overlaid message displays the objective of the room (i.e., the programming topic that will be covered with the visual elements and objects). When the user enters the room, there are one or more tokens that provide information about the theory behind the topic. This information is provided as a link to specific content (e.g., links to lecture notes, to oracle website with JAVA documentation, etc.). Additionally, an object may have a label that signifies something related to the theory (e.g., box labelled as Boolean). Furthermore, when the user is in close proximity of an object, the object may change colour if there is an interactive activity related to it. Finally, during the answering of questions, the system illuminates a green light for every correct answer and a red for incorrect ones.

### 3.6 Prototype Implementation

CodeAdventure was implemented with Unity 5.5[2] using C# scripts. It uses only unity libraries and in order to facilitate access into the scene, the majority of the developed

[2] Unity game development platform, https://unity3d.com/learn.

scripts extend monoBehavior class[3] so they can have access to the models in the scene. The monoBehavior class includes easy to use functions that facilitate seamless interaction in each frame. In order to allow for modularity, every object (e.g., lever, wheel) that interacts with the player has its own dedicated script. For the overlaid questions and information, we have utilized XML manipulation and IO libraries from the .NET framework. Finally, all room configurations, including all objects that reside inside rooms, are loaded from XML files as illustrated in Fig. 4(right) in order to allow for maintainability and expandability. The current implementation includes level one: Introduction to the JAVA programming languages and covers the following topics: Data Types, Casting, Operators and their Precedence, Expressions, Conditional Statements, and Loop Structures.

## 4 Conclusions and Future Work

In this paper, we have presented CodeAdventure, an educational game for learning and practicing introductory programming concepts. Inside CodeAdventure, players need to solve specific programming puzzles and overcome specific challenges (e.g., unlocking doors and chests), in order to explore different rooms and discover hidden items. Throughout the game, players are exposed to theory related to specific programming concepts, while at the same time they can practice and apply the concepts in a fun and engaging way. CodeAdventure incorporates different effective learning techniques, such as providing hints and clues on how to solve puzzles and providing immediate feedback on students' performance.

Since our game-based learning environment is still at a functional prototype level, we have conducted a qualitative exploratory study to evaluate its usability and accept-ability using a small group of students. In the format of focus groups and following the open-ended questions protocol, we tried to derive insights regarding students' thoughts and feelings while interacting with the environment. At this stage our main concern was to understand whether the objectives of the game were clear for them and the scenario structured and motivating. Also, special emphasis placed on the visual representation and navigation over the various elements of the game. In this respect, our understanding focused on the challenges and difficulties they had to encounter while tackling the proposed learning activities as well as on their overall experience during their interaction. Even though a detailed analysis of the data collected is still pending, the preliminary results indicate that CodeAdventure visually communicates fundamental programming concepts in a fun, stimulating and engaging manner, leading to increased student comprehension.

To further validate the impact of CodeAdventure on learning, during the next academic year (i.e., 2017/18), we plan to conduct a larger user study, using first year undergraduate computer science students taking Introduction to Programming with JAVA. In addition to the regular lecturing hours and use of provided learning material, participants will be given access to CodeAdventure as an auxiliary tool that would support their learning, will be asked to play the game on their own time and undertake particular tasks that refer to specific sections of the course. At the end of each semester, a survey questionnaire will be

[3] Creating and Using Scripts, https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html.

administered to all participants, with questions for identifying specific behavioral and usability factors such as motivation, engagement into the learning objectives, ease of use. In addition, focus groups will be organized, so as to take advantage of group interaction for acquisition of supplementary information and insights into participants' thoughts and perceptions about the game.

## References


1. Efopoulos, V., Dagdilelis, V., Evangelidis, G., Satratzemi, M.: WIPE: a programming environment for novices. In: 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2005), pp. 113–117. ACM, New York (2005). doi: 10.1145/1067445.1067479
2. Gomez, A., Santos, A., Mendes, J.A: A study on students' behaviours and attitudes towards learning to program. In: 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2012), pp. 132–137. ACM, New York (2012). doi: 10.1145/2325296.2325364
3. Hijón-Neira, R., Velázquez-Iturbide, A., Pizarro-Romero, C., Carriço, L.: Game programming for improving learning experience. In: 2014 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2014), pp. 225–230. ACM, New York (2014). doi:10.1145/2591708.2591737
4. Isomottonen, V., Nylen, A., Tirronen, V.: Writing to learn programming? A single case pilot study. In: Koli Calling, pp. 140–144. ACM, New York (2016)
5. Mach, N.: Gaming, learning 2.0, and the digital divide. In: Siemens, G., Fulford, C. (eds.) EdMedia: World Conference on Educational Media and Technology 2009, pp. 2972–2977. Association for the Advancement of Computing in Education (AACE) (2009)
6. Miljanovic, M., Bradbury, J.: Robot ON!: A serious game for improving programming comprehension. In: 5th International Workshop on Games and Software Engineering (GAS 2016), pp. 33–36. ACM, New York (2016). doi:10.1145/2896958.2896962
7. Pierce, N., Conlan, O., Wade, V.: Adaptive educational games: providing non-invasive personalized learning experiences. IEEE Computer Society (2008)
8. Prensky, M.: Digital Game-Based Learning. McGraw-Hill, New York (2001)
9. Rajala, T., Laakso, M., Kaila, E., Salakoski, T.: VILLE – a language-independent program visualization tool. In: Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), vol. 99, pp. 151–159. ACM, New York (2007)
10. Saloun, P. Velart, Z.: Adaptive hypermedia as a means for learning programming. In: Adaptive Hypermedia as a Means for Learning Programming (ICWE 2006 Workshop). ACM, New York (2006). Article 11, doi:10.1145/1149993.1150006
11. Schoeman, M., Gelderblom, H.: The effect of students' educational background and use of a program visualization tool in introductory programming. In: Annual Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT 2016). ACM, New York (2016). Article 37, doi:10.1145/2987491.2987519
12. Shuhindan, S., Hamilton, M., Souza, D.: A study of the difficulties of novice programmers. In: 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2005), pp. 14–18. ACM, New York (2005). doi:10.1145/1067445.1067453
13. Tillmann, N., de Halleux, J., Xie, T., Gulwani, S., Bishop, J.: Teaching and learning programming and software engineering via interactive gaming. In: 2013 International Conference on Software Engineering (ICSE 2013), pp. 1117–1126. ACM, New York (2013)

14. Van Eck, R.: Digital game-based learning: it's not just the digital natives who are restless. EDUCAUSE Rev. **41**(2), 16–30 (2006)
15. Wirth, M., McCuaig, J.: Making programs with the Raspberry Pi. In: Western Canadian Conference on Computing Education (WCCCE 2014). ACM, New York (2014). Article 17, doi: 10.1145/2597959.2597970
16. Wulf, T.: Constructivist approaches for teaching computer programming. In: Constructivist Approaches for Teaching Computer Programming (SIGITE 2005), pp. 245–248. ACM, New York (2005). doi:10.1145/1095714.1095771
17. Xinogalos, S., Malliarakis, C., Tsompanoudi, D., Satratzemi, M.: Microworlds, games and collaboration: three effective approaches to support novices in learning programming. In: 7th Balkan Conference on Informatics Conference (BCI 2015). ACM, New York (2015). Article 39, doi:10.1145/2801081.2801094
18. Bitter, G., Legacy, M.: Using Technology in the Classroom, 7th edn. Pearson Education Inc., Upper Saddle River (2008)
19. Bodnar, C., Anastasio, D., Enszer, J., Burkey, D.: Engineers at play: games as teaching tools for undergraduate engineering students. Res. J. Eng. Educ. **105**, 147–200 (2015)
20. Chan, E.: Motivation for mandatory courses. **7**(3) (2004). Centre for Development of Teaching and Learning
21. Eagle, M., Barnes, T.: Experimental evaluation of an educational game for improved learning in introductory computing. ACM SIGCSE Bull. **41**(1), 321–325 (2009)
22. Grissom, S., McNally, M.F., Naps, T.: Algorithm visualization in CS education: comparing levels of student engagement. In: Proceedings of the 2003 ACM Symposium on Software Visualization (2003)
23. Hijon-Neira, R., Velazquez-iturbide, A., Pizarro-Romero, C., Carrico, L.: Game programming for improving learning experience. In: The 2014 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2014), pp. 225–230. ACM, New York (2012). doi: 10.1145/2591708
24. Ireland, A., Kaufman, D., Sauvé, L.: Simulation and Advanced Gaming Environments (SAGE) for learning. In: Reeves, T., Yamashita, S. (eds.) World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, Cheasapeake, VA, pp. 2028–2036 (2006)
25. Oblinger, D.: Simulations, games and learning, pp. 1–6. EDUCAUSE Learning Initiative (2006). http://net.educause.edu/ir/library/pdf/ELI3004.pdf
26. Lincoln, Y.S., Guba, E.G.: Naturalistic Inquiry. Sage Publications, Newbury Park (1985)
27. Schwartz, P., Ogilvy, J.: The emergent paradigm: changing patterns of thought and belief (SRI International). (1979). Cited in [26]
28. Von Glasersfeld, E.: Learning as a constructive activity. In: Janvier, C. (ed.) Problems of Representation in the Teaching and Learning of Mathematics, pp. 3–18. Lawrence Erlbaum Assoc., Hillsdale (1987)
29. Denis, G., Jouvelot, P.: Motivation-driven educational game design: applying best practices to music education. Paper presented at the 2005 ACM SIGCHI international conference on advances in computer entertainment technology, Valencia, Spain (2005)