NanoDoc: Designing an Adaptive Serious Game for Programming with Working Examples Support

Pavlos Toukiloglou and Stelios Xinogalos
Department of Applied Informatics, School of Information Sciences, University of Macedonia, Thessaloniki, Greece
toukiloglou@uom.edu.gr
stelios@uom.edu.gr

Abstract: There is an increased interest in serious games about programming, particularly for younger ages. The ability to evoke motivation and retain engagement leads to better learning efficiency and a positive educational impact. Serious games usually include support systems to assist novice users. In recent years there have been attempts to enhance support with intelligent tutoring systems. These systems adapt and provide personalized learning by analyzing user behaviour. Although this is a positive evolution, support still remains in the form of hints, images and videos. The paper describes an alternative adaptive method for support with working examples. An expert solution to a similar problem is displayed to the students allowing them to address the problem and start the solving process by analogy. Our previous research showed that working examples can reduce cognitive load during problem-solving and increase the student's educational results in comparison to traditional techniques. This paper presents the design of NanoDoc, a serious game for teaching the programming concepts of sequence and iteration to elementary school students. It features adaptive support with editable and executable working examples. Each example can be examined, edited and executed freely by the user until the required knowledge level is sufficient to solve the similar problem at hand. The adaptation method is based on a Fuzzy Logic algorithm and the dynamic modelling of user knowledge level. The paper also provides some preliminary results on student usage in the classroom and a questionnaire about the game experience. Plans for future research include investigating if adaptive support with working examples can reduce cognitive load even further and how learning is affected.

Keywords: Serious games, adaptive support, programming, worked examples

1. Introduction

Computer programming has become an important skill in the 21st century and schools around the world are including it in their curriculum. Although it might differ from country to country, teaching programming commonly begins in the early grades of elementary school. Serious Games (SGs) are an essential tool for that task, particularly to novice students (Miljanovic & Bradbury, 2018). The learning mechanic is embedded with gameplay and that aspect can result in enhanced learning through motivation and active engagement (Laine & Lindberg, 2020). SGs promote student-centred learning and utilize self-teaching tools with support systems. Those systems usually include text, images and video to deliver educational content to students along with gameplay. The learning material can be received as generic as it was created for the average student and it was not fashioned for the specific needs of a user. To overcome this problem, Intelligent Tutoring Systems (ITS) have emerged to adapt the support content and provide personalized learning. ITS can act as a private tutor supervising the learning process and offering specific support when needed.

This paper presents the SG NanoDoc for teaching programming to novice users with adaptive support. The method of chosen support is Worked Examples (WEs), an approach that results in high learning efficiency and consistency according to relevant studies (McLaren et al., 2014). WEs are used in traditional teaching too, where the instructor presents a problem to the students and then guides them step by step to the solution. It is a particularly effective procedure applied to novices when they learn to solve complex problems (Sweller, 2006). Inexperienced programmers have difficulties when they encounter a new problem. They can not classify it according to the underlying principle that leads to a solution. This information is retained only by encountering several similar problems and creating a mental representation of the related programming structure that solves the category. WEs provide that information by helping students to create mental models for specific types of problems. The increased efficiency of this support method derives from the fact that by using it, students have a reduced cognitive load (Sweller, 2006). The cognitive load is the mental effort produced during learning and when it is high, the thought processes interfere. Consequently, it is essential to keep the cognitive load to a minimum, enabling students to absorb more knowledge.

In this paper, we examine a method of adaptive WE implementation in a SG about programming. NanoDoc has integrated WEs into the game, allowing players to passively study solutions to programming puzzles. We explore

the findings of a pilot use case of NanoDoc in a group of elementary school students. The results will further refine the game so it can be utilized in future studies. The next section presents a review of the related work in the field. Next, we introduce the game and its mechanics, followed by a description of the adaptive support methodology. Then, the results of the evaluation questionnaire about the game are presented. Lastly, the results are discussed and the final conclusions are addressed.

2. Related work

Although a few SGs implement adaptive support in teaching programming and prior work is scarce (Hooshyar, Malva, et al., 2021), it has been shown that it is most effective when applied. There is a diversity in the proposed methods such as Bayesian networks, Fuzzy Logic and Deep Learning. Moreover, despite the fact that the main objective is to enhance student learning, there is no common aspect of adaptation or method of presentation.

The ENGAGE game (Min et al., 2014) examines adaptive task selection and adaptive hint generation utilizing dynamic Bayesian networks to teach computer science principles to middle school students. The block-based puzzle game BOTS (Hicks et al., 2015) provides feedback in the form of personalized hints by analyzing the user output and the world state after each program execution. The HTML escape (Papadimitriou & Virvou, 2017) is an adventure game that uses Fuzzy logic to alter dynamically some of its assets like rooms, items and characters according to the player's knowledge level. AutoThinking (Hooshyar et al., 2021) employs a Bayesian network to adapt both gameplay and feedback by adjusting textual and graphical cues and adapting the difficulty of gameplay to the player's ability. In Minerva (Lindberg & Laine, 2018), a game about teaching programming to elementary school students, the content adapts to student learning styles in an attempt to increase motivation.

3. The Game

NanoDoc is a serious game designed to introduce students to the concepts of sequence and iteration in programming. It has been developed in the Unity game engine (*Unity Real-Time Development Platform | 3D, 2D VR & AR Engine*, n.d.) and deployed as a WebGL application to run on the Web. All the 3D models, textures and animations used in the game were obtained from a microgame template distributed freely from Unity Technologies. Special care has been taken to ensure that NanoDoc can run efficiently on computers with low specifications in terms of CPU and GPU. The intent was to be used in school computer labs that often are under-equipped, so techniques like lightmaps, mesh merging and occlusion culling were implemented. The game can be classified as a first-person shooter where the action and navigation in the three-dimensional space are perceived through the player's eyes (Figure 1). Students assume the role of a doctor who has been miniaturized inside a sick creature. The objective is to save the patient by curing it of the viruses while avoiding all the dangers lurking inside. A Non-Player Character (NPC) in the form of a red cell, operates as a companion to the player offering assistance and guidance.

An effort has been made to keep the quality of the game on a similar level to modern entertaining software the students use to play in their free time. The graphics are stylized and vividly colored, the hints are animated and the environment is decorated with a variety of items. The moving traps, enemies and walls, combined with music and sounds create a lively world and assist in player immersion. The mouse and keyboard inputs follow the standard consensus of the first-person shooter genre in order to minimize possible navigation control problems. We also tried to create interesting puzzles and maintain a smooth learning curve. Additionally, the adaptive support provides users WEs instead of textual instructions, decreasing cognitive load. Finally, the programming environment is block-based and built with the same functionality the students are familiar with from other educational software like Scratch.

Figure 1: The first-person shooter perspective of the game

3.1 Game mechanics

The game is a microcosm maze consisting of rooms connected with corridors and decorated with elements related to internal physiology. Several rooms include obstacles and traps whilst some of them are dead-ends. Additionally, enemies drift through the corridors, equipped with the basic behaviours of wandering, chasing and fighting. There are three types of enemies differentiated by their hit points and can be eliminated with a unique gun that players are equipped with. A handful of rooms are blocked with a substance which can only be unlocked with a corresponding protein key (Figure 2). The key is usually located in a nearby room of the sealed door and players have to solve a programming puzzle to acquire it. Each puzzle relates to a programming concept with a gradual increase in complexity as the player progresses through a total of twelve puzzles.

Figure 2: The protein key

During the programming phase (Figure 3), the game switches to a hybrid 2D/3D mode. The puzzle is depicted as a 3D grid containing a key and an avatar. In order to solve the puzzle, the player has to create a program that controls the avatar's movement in the grid and be placed in front of the key to pick it up. Some puzzles include obstacles to prevent direct access to keys, so alternative routes must be discovered. The solution algorithm is constructed in a 2D block-based programming environment. Every command or action is represented with a block that can be connected with others and form a program. The visual interface programming experience is equivalent to the Scratch educational software (Scratch - About, n.d.). In particular, blocks are coloured based on their structure collection and can be handled intuitively with drag and drop actions.

Figure 3: The programming phase

3.2 Support design

The game provides assistance during gameplay and adaptive support in programming. The NPC communicates with text messages and informs players about upcoming dangers or instructs them on game controls and mechanics. Those messages are animated and placed in the lower centre of the screen in order to grab the player's attention. In addition, flashing arrows in specific corridors and a mini-map are offered to assist with navigation in the maze. The adaptive instructional support is activated within the programming phase. The game creates a model of player knowledge level for the domains of sequence and iteration and updates it with every submitted solution. If an insufficiency of the required knowledge for the current puzzle is detected, the game presents players with a worked example of a compatible problem. The WE is editable and players can examine and execute it without restrictions until they conclude that the required knowledge was perceived (Figure 4). Whenever the adaptive support is triggered for a puzzle, an associated animated button placed in the toolbar area is enabled. With this button, players can swap between their code/puzzle pair and the WE ones without restraints until they solve the puzzle.

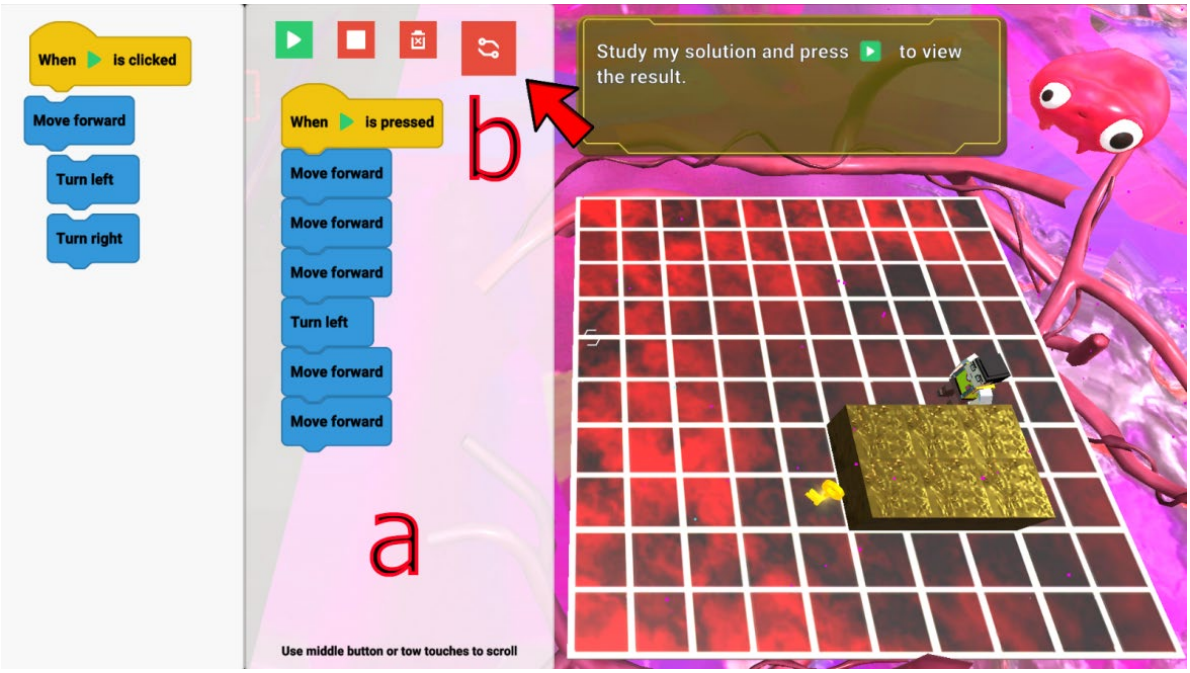

Figure 4: Support enabled (a) the WE (b) The hide WE button

4. Adaptive support

The adaptive support model extends a previous method from the education game FuzzEg (Papadimitriou et al., 2019) and applies Fuzzy logic to describe student knowledge level in a domain. The process can be defined by five steps:

1. Gather learning data from the player's submitted solution
2. Fuzzify the input values with membership functions
3. Apply a rule set to the fuzzy output
4. Map the fuzzy output into a crisp value
5. Adapt support according to the crisp values

Each time the player submits a puzzle solution, the game estimates a value that represents how optimized the solution is in terms of programming structures utilization. The equation (Equation 1) is based on a previous study (Toukiloglou & Xinogalos, 2022) and calculates learning efficiency by dividing the number of blocks of the optimal solution by the number of blocks of the solution committed by the student. The coefficient D defines the puzzle difficulty and varies according to game design. However, since every puzzle in this game uses the accumulated knowledge of the previous ones, the overall difficulty remains steady and D becomes a constant equal to 1.

$$Efficiency = \frac{Optimal\ Solution}{Student\ Solution} \times D$$

Equation 1: Calculation of learning efficiency

The formula transforms the data from a submitted solution into a positive decimal number with a maximum value of 1 which is the best possible result of a solved puzzle. The following step converts the efficiency score to fuzzy values using four membership functions (Papadimitriou et al., 2019) as depicted in Figure 5.

$$\mu Un(x) = \begin{cases} 1, & x < 45 \\ 1 - \frac{x-45}{5}, & 45 < x < 50 \\ 0, & x \geq 50 \end{cases}$$

$$\mu Uk(x) = \begin{cases} \frac{x-45}{5}, & 45 < x < 50 \\ 1, & 50 \leq x \leq 65 \\ 1 - \frac{x-65}{5}, & 65 < x < 70 \\ 0, & x \leq 45, x \geq 70 \end{cases}$$

$$\mu K(x) = \begin{cases} \frac{x-65}{5}, & 65 < x < 70 \\ 1, & 70 \leq x \leq 85 \\ 1 - \frac{x-85}{5}, & 85 < x < 90 \\ 0, & x \leq 65, x \geq 90 \end{cases}$$

$$\mu L(x) = \begin{cases} \frac{x-85}{5}, & 85 < x < 90 \\ 1, & 90 \leq x \leq 100 \\ 0, & x \leq 85 \end{cases}$$

Figure 5: Membership functions of knowledge fuzzy sets

Every puzzle of the game is associated with a specific learning domain, so the fuzzy output describes the knowledge level of the student in that domain. The knowledge level is described as a quadruplet (μUn, μUK, μK, μL) of the following fuzzy sets (Papadimitriou et al., 2019):

1. Unknown (Un): The domain concept is unknown to the player. Efficiency score is from 0.0 to 0.5
2. Unsatisfactorily Known (Uk): Player has some knowledge of the concept. Efficiency score is from 0.45 to 0.7
3. Known (K): Player knows the concept sufficiently. Efficiency score is from 0.65 to 0.9
4. Learned (L): Player knows the concept completely. Efficiency score is from 0.85 to 1.0

The fuzzy set definitions enforce certain restrictions for each quadruplet. The value of each component is ranging between 0 and 1 and the total sum of all the components is always equal to 1. Moreover, the values are adjoined since it is illogical to skip an intermediate set. Next, the defuzzify process converts the output data to a crisp value that defines the whole set. This value represents the knowledge level of the domain that is reviewed and is calculated using the Maximum-Membership method where the maximum value of the set is chosen. In the final step, a decision is made upon providing support to players according to a ruleset as shown in Table 1. This ruleset is based on the knowledge level for each domain which is constantly updated with every puzzle and the number of failed attempts of the current problem. Finally, support can also be triggered automatically by the

system when it detects inactivity for at least 120 seconds. This additional measure was set to address cases where students are stumped by the problem and can not continue without help.

Table 1: Support activation ruleset

Knowledge domain level	Failed attempts
Learned	4
Known	3
Unsatisfactorily Known, Unknown	2

5. Evaluation

To evaluate the state of the game regarding playability and support functionality, a questionnaire has been conducted in January of 2021 and lasted two weeks. A group of 85 (41 male, 44 female) elementary students of the same school, aged 10-12 volunteered to play the game. Most of the students (82%) had some programming experience with Scratch (*Scratch - About*, n.d.) and the rest were complete novices. The procedure was supervised by a teacher and took place in the school's computer lab. After a play session that lasted 45 minutes, students anonymously answered an online questionnaire that consisted of 3 sections with Likert scale responses. The first section is a single question querying the overall attitude of the student towards learning programming with serious games. The next two sections include several sub-questions and investigate student perception of the game and the difficulties they encounter. Lastly, a non-mandatory question was granted to comment openly about the game and it was answered by 31% of the students. The vast majority of comments were positive whereas some were requests for new features like a larger map, adding new weapons or a multiplayer option (Table 2). The rest of the results are shown as percentages in Figures 6 and frequencies in Figures 7 and 8.

Table 2: A sample of student comments (translated from Greek)

Player	Comment
Student 1	“The game had nice 3D graphics. I wanted to last longer and be more difficult”
Student 2	“The game was a lot of fun”
Student 3	“I wanted more weapon options and better enemies”
Student 4	“The game was nice and I want to replay it again”
Student 5	“The game was a blast. Add a multiplayer option please”
Student 6	“I enjoyed learning programming with this”

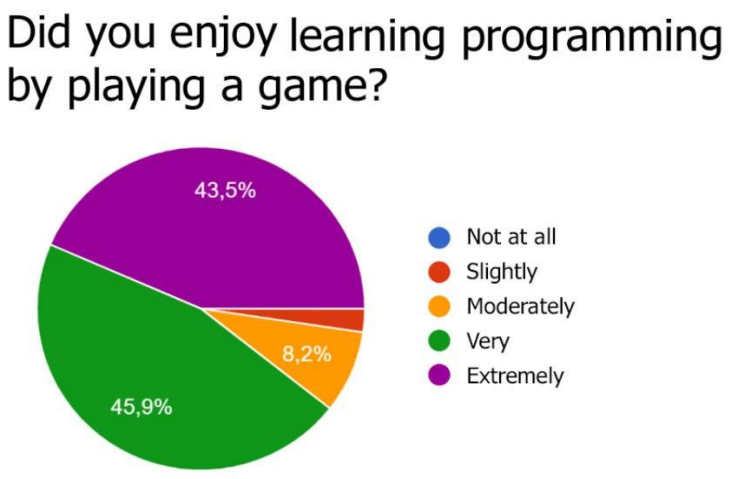

Figure 6: First section results (translated from Greek)

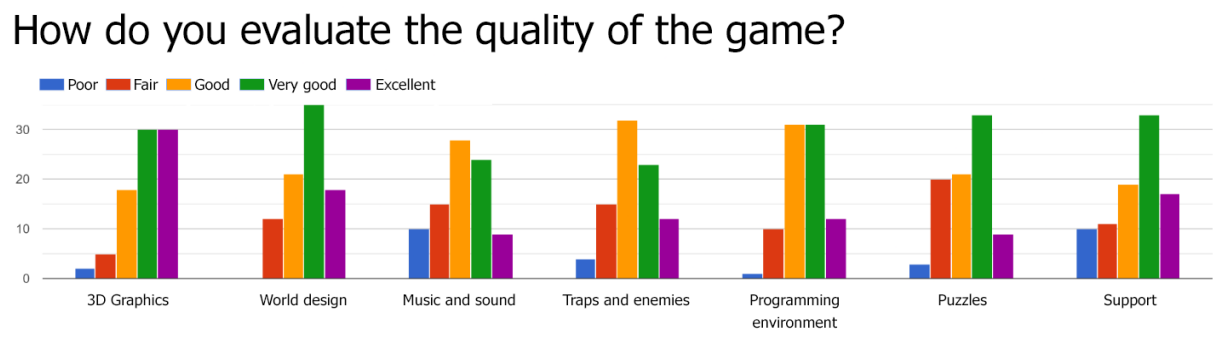

Figure 7: Quality results (translated from Greek)

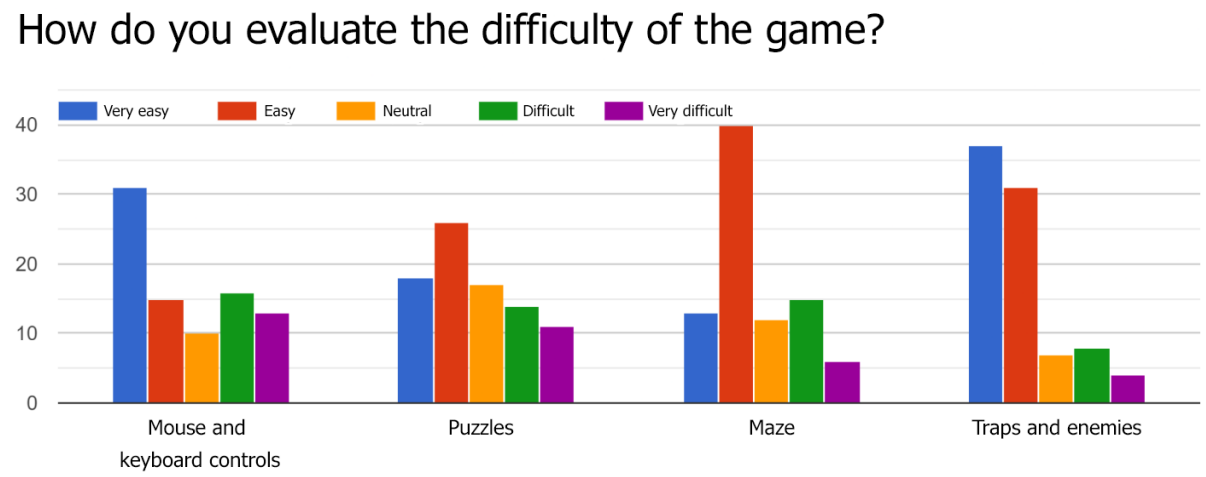

Figure 8: Difficulty results (translated from Greek)

6. Discussion

Although the number of participants in the questionnaire was small, the results were very encouraging. The answers of the first section concluded with an almost 90% positive opinion about learning programming with games. This finding agrees with previous studies about the motivation potential of serious games in education (Laine & Lindberg, 2020). The second section questioned the quality of various aspects of the game. The 3D visual representation of the world proved to be one of its strong assets and received good reviews both on graphics and world design. Sound and music however had a moderate impact, possibly due to external factors. During the game session in the computer lab, students were asked to lower the sound volume on their computers. This was decided to reduce the overall volume of the class and mitigate the disturbance of the thinking process during puzzle solving. However, this could have been avoided if students had received headphones. Any future designs that involve sound and music in serious games should take that factor into consideration.

The majority of the students had used a block-based programming environment in the past with Scratch. The resemblance with NanoDoc's programming environment proved to be beneficial as they reported no problems with the interface. It should be noted that half of the puzzles involved the iteration structure. Nonetheless, 85% of the students were only taught the sequence structure in the school while the rest were complete novices. Despite that, they liked the puzzles and found them relatively easy to solve. Also, the support method received very positive results. This concludes that the support method was effective in teaching the programming structures when needed and students manage to solve those puzzles without external help.

Enemies and traps in the game were relatively easy for the experienced players. The behaviours of enemies were simplistic and bumping into a trap would only cause a delay without other consequences. A better approach would have been to implement a Dynamic Difficulty Adjustment (DDA) (Min et al., 2014) system and alter the difficulty in compliance with the player's skill, maintaining the level of challenge. Nevertheless, dynamic difficulty was not in the scope of the study so an easy level of difficulty in obstacles was decided to help less experienced

players. This design decision caused a moderate response to the related question since it emerged that most of the students were experienced players in commercial games. Additionally, the answers in the third section about the difficulty evaluation in those areas of the game showed the same results.

Continuing with the third section of the questionnaire, students reported no problems with the game controls. The NPC companion provided useful tips about keyboard and mouse usage in the first parts of the game. Nonetheless, the game followed the typical mapping of functions found in similar games. This consistency allowed experienced players to immediately familiarize themselves with the controls and remove a potential side problem. The controls standardization is a common practice in commercial games and it is strongly advised to be followed as it was proven to be valid.

Despite the fact that the maze was difficult to navigate including close loops, dead-ends and split paths, students reported no problems. NanoDoc provided a mini-map and visual aid with flashing arrows pointing in the right direction at specific points in the maze. Players tend to dismiss text messages or alert boxes while navigating the world as they consider them to be annoying interruptions. Presenting the intended information as part of the game world confirmed that practice and will be followed in the future.

7. Conclusions

The serious game NanoDoc is presented with the intent of teaching programming to novice students. The game features an adaptive support system based on the Fuzzy logic model. The game creates a student model and provides support with working examples when it detects inefficiency in a knowledge domain. An evaluation of the game was conducted to a group of elementary students with a questionnaire. The results suggest an overall positive opinion about the game elements and mechanics. However the areas of difficulty and sound require refinement to meet student expectations. Enemies and obstacles should provide a greater challenge to keep player retention. It is also concluded that the adaptive algorithm operates as intended and students manage to solve problems without external help in unknown knowledge domains. These findings suggest that the problematic mentioned areas should be improved in the next version of the game.

In previous work (Toukiloglou & Xinogalos, 2022), we compared WEs with textual support and showed that WEs provide better learning efficiency due to reduced cognitive load. We plan to use NanoDoc in an empirical study that will investigate if the cognitive load can be reduced even further. An adaptive worked example support version of the game will be compared with a non-adaptive one in terms of learning efficiency. The research will be conducted on elementary students with limited previous experience in programming.

Acknowledgment

This work is part of a project that has received funding from the Research Committee of the University of Macedonia under the Basic Research 2020-21 funding programme.

References

Hicks, D., Dong, Y., Zhi, R., Cateté, V., & Barnes, T. (2015). BOTS: Selecting next-steps from player traces in a puzzle game. *CEUR Workshop Proceedings*, *1446*.

Hooshyar, D., Malva, L., Yang, Y., Pedaste, M., Wang, M., & Lim, H. (2021). An adaptive educational computer game: Effects on students' knowledge and learning attitude in computational thinking. *Computers in Human Behavior*, *114*(September 2020), 106575. https://doi.org/10.1016/j.chb.2020.106575

Hooshyar, D., Pedaste, M., Yang, Y., Malva, L., Hwang, G. J., Wang, M., Lim, H., & Delev, D. (2021). From Gaming to Computational Thinking: An Adaptive Educational Computer Game-Based Learning Approach. *Journal of Educational Computing Research*, *59*(3), 383–409. https://doi.org/10.1177/0735633120965919

Laine, T. H., & Lindberg, R. S. N. (2020). Designing Engaging Games for Education: A Systematic Literature Review on Game Motivators and Design Principles. *IEEE Transactions on Learning Technologies*, *13*(4), 804–821. https://doi.org/10.1109/TLT.2020.3018503

Lindberg, R. S. N., & Laine, T. H. (2018). Formative evaluation of an adaptive game for engaging learners of programming concepts in K-12. *International Journal of Serious Games*, *5*(2), 3–24. https://doi.org/10.17083/ijsg.v5i2.220

McLaren, B. M., van Gog, T., Ganoe, C., Yaron, D., & Karabinos, M. (2014). Exploring the Assistance Dilemma: Comparing Instructional Support in Examples and Problems. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Intelligent Tutoring Systems* (Vol. 8474, pp. 354–361). Springer International Publishing. https://doi.org/10.1007/978-3-319-07221-0_44

Miljanovic, M. A., & Bradbury, J. S. (2018). A Review of Serious Games for Programming. In S. Göbel, A. Garcia-Agundez, T. Tregel, M. Ma, J. Baalsrud Hauge, M. Oliveira, T. Marsh, & P. Caserman (Eds.), *Serious Games* (Vol. 11243, pp. 204–216). Springer International Publishing. https://doi.org/10.1007/978-3-030-02762-9_21

Min, W., Mott, B., & Lester, J. C. (2014). Adaptive Scaffolding in an Intelligent Game-Based Learning Environment for Computer Science. *The Second Workshop on AI-Supported Education for Computer Science*, 41–50.

Papadimitriou, S., Chrysafiadi, K., & Virvou, M. (2019). FuzzEG: Fuzzy logic for adaptive scenarios in an educational adventure game. *Multimed Tools Appl*, *78*(22), 32023–32053. https://doi.org/10.1007/s11042-019-07955-w

Papadimitriou, S., & Virvou, M. (2017). Adaptivity in scenarios in an educational adventure game. *2017 8th International Conference on Information, Intelligence, Systems and Applications, IISA 2017*, *2018-Janua*, 1–6. https://doi.org/10.1109/IISA.2017.8316453

Scratch—About. (n.d.). Retrieved 25 November 2021, from https://scratch.mit.edu/about

Sweller, J. (2006). The worked example effect and human cognition. *Learning and Instruction*, *16*(2), 165–169. https://doi.org/10.1016/j.learninstruc.2006.02.005

Toukiloglou, P., & Xinogalos, S. (2022). Ingame Worked Examples Support as an Alternative to Textual Instructions in Serious Games About Programming. *Journal of Educational Computing Research*, 073563312110736. https://doi.org/10.1177/07356331211073655

Unity Real-Time Development Platform | 3D, 2D VR & AR Engine. (n.d.). Retrieved 12 December 2021, from https://unity.com/