Education and Information Technologies
https://doi.org/10.1007/s10639-023-12317-z

Evaluating support systems and interface efficiency in Hour of Code's Minecraft Adventurer

Pavlos Toukiloglou[1] · **Stelios Xinogalos**[1]

Received: 31 March 2023 / Accepted: 27 October 2023

Abstract
Hour of Code is a widely recognized global event that aims to introduce programming to novice users and integrate computer science into education. This paper presents an analysis of the effectiveness of the support system and user interface of Minecraft Adventurer, a serious game designed for the Hour of Code global event. Although previous studies have primarily focused on the educational benefits of Hour of Code games, there has been limited research on their support methods. Therefore, this paper aims to address this gap with an empirical study of the experience of 104 students who played the game for one hour. Student progress was tracked by an administering teacher and after the game session, a questionnaire was administered to collect data on the participant's perceptions of the support system, interface efficiency, and overall experience with Hour of Code. The results of the study reveal significant problems with the aforementioned systems, which apply not only to Minecraft Adventurer but also to several other similar serious games. Additionally, the findings showed a correlation between the utilization of the support system and student performance, indicating that student's comprehension of the support system significantly influences their learning outcomes. This paper concludes by providing potential solutions to address the identified insufficiencies, offering valuable insights for future researchers and game developers on the design and evaluation of serious games for educational purposes.

Keywords Hour of Code · Serious games · Programming · Support · User interface

✉ Stelios Xinogalos
stelios@uom.edu.gr

Pavlos Toukiloglou
toukiloglou@uom.edu.gr

[1] Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece

Published online: 04 December 2023

1 Introduction

The Hour of Code (HoC) is a global event that aims to promote computer science education by introducing millions of students to the world of coding and programming. It takes place annually and is organized by Code.org, a non-profit organization dedicated to improving access to computer science education. The HoC is designed to be an easy and enjoyable way for students of all ages and skill levels to learn the fundamentals of coding. Activities are conducted in over 160 (Code.Org 2021 Annual Report, n.d.) countries and typically involve completing online tutorials or challenges that teach coding concepts and programming languages, such as Scratch, HTML, and Python. The HoC (Hour of Code Activities, n.d.) website provides a diverse range of resources, tutorials, and activities that can be used to teach programming. These activities are engaging and interactive, using games, animations, and simulations to make the programming concepts more relatable and understandable to students. Additionally, in order to capture students' interest and keep them engaged, many activities are based on popular commercial games or movie themes.

The majority of HoC games have a common design and code base, which results in easier development and maintenance. As a consequence, users who are familiar with one game can easily adapt to similar designs, reducing the learning curve for each new game. Most of the HoC games include a block-based programming environment that is aimed to be simple and intuitive to novice programmers. The code is presented visually and programs are constructed by dragging and dropping blocks that represent various commands. These blocks are usually color-coded and snap together to help users create syntactically correct code. The similarity in design continues to support systems and user interface (UI) which also means that possible insufficiencies are shared with the rest of the games. The support system is an essential element of a serious game (SG) as it provides assistance and guidance when users encounter difficulties or errors. In HoC games support systems take many forms such as tutorials, step-by-step instructions, hints and tips, error messages, and feedback on completed tasks.

User Experience (UX) plays a crucial role in the development of SGs as it significantly influences the acceptability of digital games (Moizer et al., 2019). UX according to The International Standard on Ergonomics of Human System Interaction is defined as *"a person's perceptions and responses that result from the use or anticipated use of a product, system, or service"* (DIS, 2009). The research in UX evaluation for SGs (Martinez et al., 2022; Abdellatif et al., 2018) reveals that UI is a key characteristic and an important factor of user learning. UI refers to the in-game methods of interaction between the users and the SG. The UI implementation in terms of accessibility, usability, and comprehensibility is associated with the success of SGs (Lanyi et al., 2012; Mikovec et al., 2009). Therefore, UI design guidelines have been devised by researchers and organizations that include best practices and principles for developers (Johnson, 2020; Apple, 2022). Given that support methods are implemented through the UI, the design choices of HoC UI elements will be examined to determine their relation with student performance.

Table 1 Hour of Code games with similar support systems and UI design with Minecraft Adventurer

Game	Link
Minecraft Designer	https://studio.code.org/s/minecraft/lessons/1/levels/1
Minecraft Hero's journey	https://studio.code.org/s/hero/lessons/1/levels/1
Minecraft Voyage Aquatic	https://studio.code.org/s/aquatic/lessons/1/levels/1
Frozen - Anna and Elsa	https://studio.code.org/s/frozen/lessons/1/levels/1
Star Wars - Building a galaxy	https://studio.code.org/s/starwarsblocks/lessons/1/levels/1
Artist	https://studio.code.org/s/artist/lessons/1/levels/1
Flappy code	https://studio.code.org/flappy/1
Dance party	https://studio.code.org/s/dance-2019/lessons/1/levels/1
Disney Infinity play lab	https://studio.code.org/s/infinity/lessons/1/levels/1
Angry Birds	https://studio.code.org/hoc/1
Outbreak simulator	https://studio.code.org/s/outbreak/lessons/1/levels/1

This study aims to investigate the implementation of the support system and user interface in the HoC game Minecraft Adventurer. We argue that the effectiveness of these systems directly affects the user's learning efficiency and performance. Minecraft Adventurer shares similar support systems and UI features with many other HoC games. At the time of writing, eleven games with identical characteristics in terms of support systems and UI design with the aforementioned title can be found in HoC and are presented in Table 1. It is worth noting that this support approach is employed in many other SGs about programming (Tynker, n.d.; Roboblocky, n.d.), besides the HoC games. Therefore, the findings of this study have implications beyond the HoC and can contribute to the broader field of SG for education.

More specifically, the study will explore the following research questions:

RQ1: How do novice users perceive the support systems and UI of the HoC game Minecraft Adventurer?
RQ2: To what extent do the design of the support system and UI of the HoC game Minecraft Adventurer affect game performance?
RQ3: How can the support system and UI featured in HoC games be improved in terms of learning efficiency?

The subsequent sections of this paper are organized as follows: Section 2 provides a thorough depiction of the Minecraft Adventures game, followed by a review of the current state of the art in the field in Section 3. In Section 4, we elaborate on the study methodology used in this research, which leads to the presentation of the findings in Section 5. Section 6 includes a detailed discussion of the results obtained along with the limitations of the research.

2 Minecraft Adventurer

All the HoC games follow a design approach that is heavily visual in nature, both during the solution construction and result display phases. Visual environments can increase engagement in novice users as they demonstrate skills and strategies in a

familiar or easily understood context. As the title of the game implies, it is based on the commercial game Minecraft and themed accordingly in terms of graphics, music, enemies, and available actions. Minecraft is a popular sandbox video game (Minecraft Player Count & Stats, 2023 that allows players to build and explore virtual worlds made of blocks. It is known for its open-ended gameplay and creative building elements, which allow players to create and manipulate their own virtual world. Minecraft Adventurer introduces students to basic programming concepts by allowing them to navigate, mine, craft, manipulate, and explore in a 2D world using block-based coding. In block-based programming, code instructions are represented with visual blocks rather than text as in traditional programming languages. This method of coding offers a more intuitive and user-friendly way for novice programmers to learn and create basic programs.

The game interface includes a game world representation, an instruction panel, and a right-side area with a "toolbox" and "workspace" (Fig. 1). Students at the start of the game select an avatar and their programs control the avatar's actions within the world. The game world consists of various objects such as trees, rocks, animals, etc., forming puzzles students need to solve by completing the level's objectives. Programs are created by assembling in sequence blocks listed in the toolbox. Each block is dragged and dropped in the workspace area and needs to be connected with the rest of the program to function. The toolbox includes commands for movement control or avatar actions such as placing or destroying objects. Additionally, the toolbox contains blocks for iteration and conditional statements that are adapted to facilitate the puzzle requirements. A horizontal line with 14 nodes is placed in the upper part of the screen displaying the progress of the player (Fig. 1d). Each node represents a level of the game and it is painted with different colors depending on the solution state. Specifically, the dark green color signifies puzzles that have been successfully solved with an optimal solution in terms of the number of blocks employed, the light green color represents puzzles with non-optimal solutions and

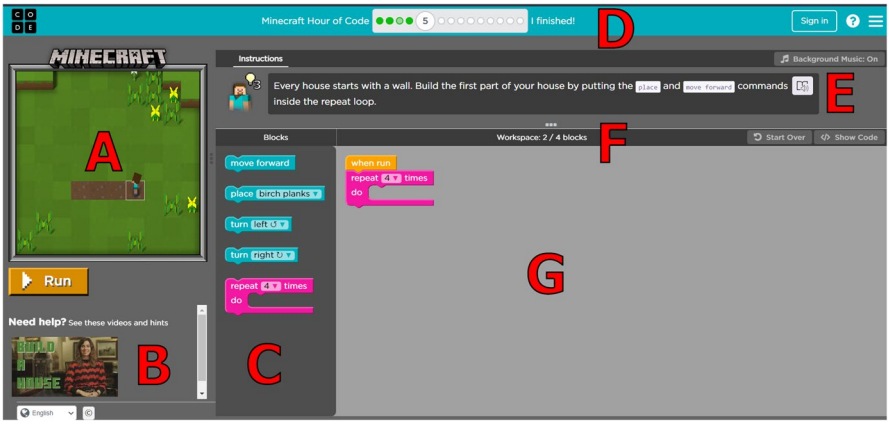

Fig. 1 The Minecraft Adventurer interface. **a** The game world, **b** The video support, **c** The block commands, **d** The progress indicator, **e** The tips support, **f** The block counter, and (**g**) The programming workspace

white indicates puzzles that have yet to be solved. Moreover, it should be noted that despite the presence of 14 nodes, the actual number of puzzles amounts to 13. This distinction arises from the fact that the final level serves as a free level, allowing players unrestricted utilization of the block commands, thereby deviating from the conventional puzzle-solving context.

The game's design allows students to relate their programs with the actions taking place in the game world by observing the step-by-step command execution. During the coding phase, the game provides users with tips based on the selected instruction mode. Additionally, after submitting a solution, a pop-up window displays the corresponding JavaScript code (Fig. 2). This feature allows students to familiarize themselves with the actual code generated and gain a better understanding of the programming concepts being taught. Depending on the solution's efficiency, the game prompts users to either proceed to the next level or attempt to solve the problem again using fewer blocks. Minecraft adventures incorporate story and narrative elements to engage students. The scenario recounts the hero's adventures to gather the necessary resources in order to build a house. The game also includes audio elements with background music and occasional effects to emphasize events. Throughout the course of the adventure, players encounter puzzles that require the application of specific programming structures, which are presented for each level in Table 2. The instructional approach of the game focuses on a structured presentation of programming concepts, with the sequence being the initial and fundamental structure taught. The iteration follows as a solution to large sequential programs and

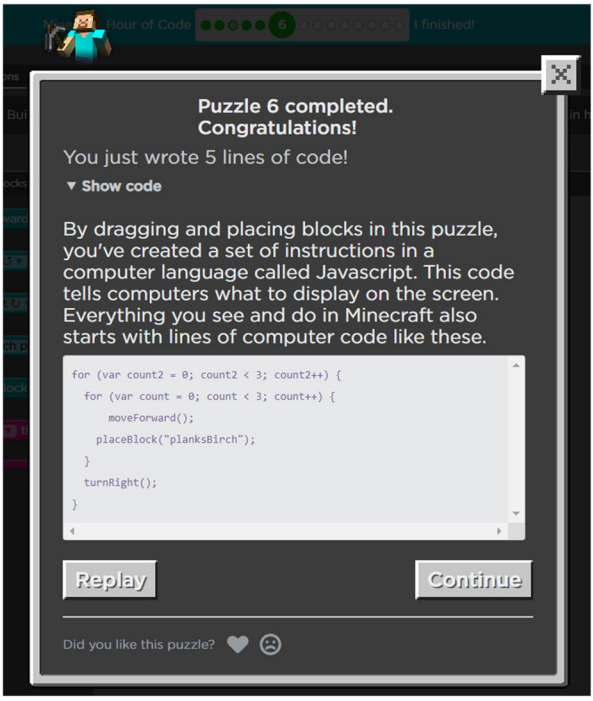

Fig. 2 A successful level completion. The total lines of code and a snippet of javascript are displayed

Table 2 Programming structure taught per level

Programming structure	Puzzle level												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Sequence	X	X	X	X	X	X	X	X	X	X	X	X	X
Iteration					X	X	X	X	X	X	X	X	X
Condition											X	X	

nested loops for more complex problems. The condition domain is presented in the final stages and is integrated with the previously taught concepts to address more challenging programming puzzles.

3 Related work

A recent systematic review of the HoC (Yauney et al., 2021) has revealed that the reported research had limitations regarding the provided details on participants and results. While a considerable amount of research has been published on HoC, only a small portion of it is based on experiments, particularly in K-12 schools. Additionally, the research is dispersed across multiple sources, and there have been no efforts to synthesize it. Yauney et al. (2021) suggest that research into the impact of HoC on student interest in computer science and perceptions of the field is essential. Such research could provide evidence of HoC's value and areas for improvement.

GhasemAghaei et al. (2016) conducted an analysis of the HoC Minecraft game utilizing affective walkthrough and affective heuristic evaluation techniques. Although the overall assessment was positive, one of the areas that have been identified for improvement was the support system. It was suggested to add hints for informing the user when a shorter solution than that submitted by him/her is feasible. Researchers examined the HoC support improvement through autogenerated hints. Buwalda et al. (2018) developed a heuristics-based approach for providing next-step hints in HoCs puzzles. They demonstrated that their hints can effectively simulate the corresponding expert's level without utilizing large quantities of student data. Piech et al. (2015) explored the use of historical student data to autonomously generate hints for the HoC games. The paper suggests that an effective hint generation model should have as a basis of prediction the way an expert teacher would encourage a student to make forward progress.

The HoC activities are characterized by their static and linear nature, providing a uniform sequence of tasks to all students, regardless of their prior skills and learning speed. As a result, there has been considerable research aimed at enhancing the provided support. One such approach is presented by Effenberger (Effenberger, 2019; Effenberger & Pelánek, 2018), which involves an adaptive method that personalizes game tasks based on student performance evaluation. The system evaluates a student's past performance and recommends activities of appropriate difficulty levels to optimize their learning experience. Basawapatna et al. (2019) propose a new

approach to HoC tutorials based on a differentiated instruction strategy guided by the Zones of Proximal Flow (ZPF) framework. The ZPF tutorials offer a navigation structure that allows users to choose appropriate detail based on their self-assessed state of flow. The authors conclude that their ZPF tutorial outperformed the tutorial in HoC style in terms of student retention and motivation.

The aforementioned research establishes that a significant number of researchers hold the view that the method of support for HoC necessitates enhancement. Our study presents substantiating evidence for this assertion through an analysis of questionnaire data and students' in-game performance.

4 Study methodology

4.1 Study design

The study aimed to investigate the effects of support systems in the HoC serious game Minecraft Adventurer. Additionally, the study investigates the interface efficiency of the serious game in communicating the intended information to its users. The game theme and difficulty setting target an audience that is familiar with Minecraft and novice in programming. The research was based on a quasi-experimental design with a single group of students from two elementary schools. The game did not provide a mode where the support methods could be disabled thus it was not possible to create a control group of students for the experiment. All the support system elements were seamlessly integrated into the UI and gameplay, precluding the possibility of avoidance or disregard even under the hypothetical circumstance where students were specifically instructed to do so.

Upon completion of the game, students answer an anonymous questionnaire that collects information on students' perceptions of the game's support system, interface efficiency, and their overall experience with the HoC event. Given the relatively young age of the students, particular care was exercised to ensure that the questionnaire remained straightforward and without excessive technical terminology. Consequently, the questionnaire was intentionally designed to use simple language and was completed in a short period of time.

4.2 Procedure

The study was carried out in the context of a computer science course during the yearly HoC event in December 2022. Before the experiment, students were briefed about the event and the benefits of learning programming and analytical thinking. A demonstration was performed to show the students how to program in a block-based environment and to present the main elements of the game. No instructions on the specific features of the support system or a detailed explanation of all the provided interface components were given.

The HoC session was conducted in the school computer lab for 45 minutes. During this period, students were instructed to rely solely on the game's support system.

Each was sitting alone at a workstation and was not allowed to talk or interact with the rest of the class. The session was administered by a teacher who was not authorized to guide or advise on puzzle solutions but only to assist in case of a technical problem. Students were informed that both their performance and their responses to the subsequent questionnaire would be anonymously registered and the collected data would be processed for the purpose of study. Furthermore, they were granted the option to discontinue the procedure at their discretion.

After the end of a game session, students anonymously answered an online questionnaire that consisted of 4 sections. Although the questionnaire was anonymous, students registered their class and school data. Moreover, the questionnaire form had an automatic timestamp, registering the time of response submission. Upon completion of the questionnaire, the administrative teacher registered the game results for each student in a spreadsheet along with their respective class and timestamp. Specifically, the teacher collected the total number of puzzles solved and the number of puzzles solved with more lines of code than allowed/proposed. Combining the teacher's results spreadsheet sequence, the timestamp data, and class information within the questionnaire dataset, it was feasible to correlate student performance and their respective responses while keeping the data anonymous.

4.3 Participants

The focus of the game on supporting novices in learning the basic concepts of programming and its design to appeal to younger ages made primary school students the ideal participants for the study. The inclusion criteria for the participants of the study were: (1) being an elementary school student, (2) being a novice in programming, and (3) following the Greek school curriculum in computer science at the time of the study. The exclusion criteria were: (1) having prior experience with programming, (2) having previously played Minecraft Adventurer or similar HoC games. The participants were recruited from two schools located in the same geographical area and socioeconomic status. The sample consisted of 104 participants (42.3% female and 57.7% male students) with an age range of 10 to 12 years old.

4.4 Data collection

The study data was obtained anonymously from two sources as already mentioned. The first data source was a questionnaire, as shown in Table 3. The second data source referred to metrics collected by the administrative teacher manually in a spreadsheet and contained the results of each game level in terms of solution efficiency. An example of that data is shown in Table 4 where each column represents a different student. The first two lines store the total number of levels solved and the number of levels solved inefficiently, whereas the third line calculates the efficient solutions. According to the game rules, an inefficient solution is defined as a solution that completes the level objective with a greater number of blocks than optimal. In most cases, it referred to solutions that required the use of an iteration structure and students used a sequential one instead. Finally, questionnaires included a free

Table 3 The student questionnaire (Q1-Q3 and Q6-Q10 have the same responses)

ID	Question
Q1	How helpful were the text instructions in solving the puzzles? a) Not at all b) A little c) Moderately d) Very Much e) Extremely
Q2	How helpful were the video presentations in solving the puzzles?
Q3	How helpful was the display of the number of blocks for the best solution in the workspace?
Q4	What instruction mode did you use most frequently? a) Less b) More c) I did not know that such an option existed
Q5	What was your reaction when you were prompted that it is possible to complete the level with fewer lines of code? a) I pressed "continue" and moved on to the next level b) I pressed "try again" and tried to solve it with fewer blocks c) I did not understand or read the message. I ignored it and pressed "continue"
Q6	Did you like the game theme?
Q7	Did you like the game graphics?
Q8	Did you like the music?
Q9	Did you like the programming puzzles?
Q10	Did you like the block-based programming style?
Q11	Would you like to try another Hour of Code game? a) Yes b) No

Table 4 An example of student progress table

Student ID	1	2	3	4	5	6	7	8	9	10	11	12
Total levels solved	12	12	11	11	9	9	11	12	9	9	11	11
Inefficient solutions	5	4	4	6	6	6	6	7	3	4	7	4
Efficient solutions	7	8	7	5	3	3	5	5	6	5	4	7

comment section where students provided feedback on a free form about their experience with the game, programming, and HoC.

We employed Cronbach's alpha as a method to assess the reliability of the questionnaire items and evaluate the internal consistency among them. Three items (questions 4, 5, and 11) were excluded because they did not measure student opinions related to support systems and game characteristics, in addition to their use of different scale items. The computed Cronbach's alpha coefficient resulted in a value of $\alpha = 0.67$, which is considered an acceptable level of reliability. This result suggests that the questionnaire retains a sufficient degree of internal consistency, making it a reliable tool for data collection in the context of our research objectives.

5 Data analysis and results

Table 5 presents the descriptive statistics of the questionnaire on the student responses according to the question IDs introduced in Table 3. The first section of the questionnaire refers to questions about the effectiveness of the support system and the methods used to display and propagate information to students. The results of the question (Q1), "How helpful were the text instructions in solving the puzzles?" indicate that most students found the text instructions to be moderately helpful (20.19%). However, a significant portion of students found them either not helpful (21.15%) or only a little helpful (25%). Only 18.27% considered them very helpful, and 15.38% considered them extremely helpful. The text instructions were located above the workspace and provided tips on solving the puzzle and using commands (Fig. 1e). The second question (Q2) was focused on evaluating the efficiency of the video presentations offered to students in levels 1, 5, and 9. The video presentations were designed to introduce new concepts and were placed in the lower left corner of the screen for optional re-watch after being closed, as shown in Fig. 1b. The results indicated that the majority of students found the video presentations to be either unhelpful (70.19%) or only a little helpful (8.65%). The third question (Q3) was focused on evaluating the helpfulness of the block counter that is placed above the workspace, as shown in Fig. 1f. The counter displays the number of blocks for the best solution in the current puzzle and the number of blocks for the ongoing solution. It is designed to serve as an indicator for students to assess their solution progress. The results show that 31.73% of students found the block counter to be unhelpful or only a little helpful (26.92%). On the other hand, 20.19% of students found the block counter to be extremely helpful, while 11.54% found it moderately helpful and 9.62% found it very helpful.

The fourth question (Q4) aimed to determine the usage of the scaffolded hint functionality. The game provides text instructions that are scaffolded in two or three levels depending on the complexity of the puzzle. If the student is struggling to solve

Table 5 Questionnaire descriptive statistics

ID	A[1]	B[1]	C[1]	D[1]	E[1]
Q1	22 (21.15%)	26 (25%)	21 (20.19%)	19 (18.27%)	16 (15.38%)
Q2	73 (70.19%)	9 (8.65%)	10 (9.62%)	4 (3.85%)	8 (7.69%)
Q3	33 (31.73%)	28 (26.92%)	12 (11.54%)	10 (9.62%)	21 (20.19%)
Q4	32 (30.77%)	17 (16.35%)	55 (52.88%)	–	–
Q5	80 (76.92%)	16 (15.38%)	8 (7.69%)	–	–
Q6	7 (6.73%)	7 (6.73%)	25 (24.04%)	14 (13.46%)	51 (49.04%)
Q7	12 (11.54%)	20 (19.23%)	22 (21.15%)	21 (20.19%)	29 (27.88%)
Q8	30 (28.85%)	20 (19.23%)	20 (19.23%)	11 (10.58%)	23 (30.77%)
Q9	15 (14.42%)	20 (19.23%)	24 (23.08%)	13 (12.50%)	32 (30.77%)
Q10	10 (9.62%)	19 (18.27%)	26 (25.00%)	15 (14.42%)	34 (32.69%)
Q11	74 (71.15%)	30 (28.85%)	–	–	–

[1]Student responses (frequency/percentage)

a puzzle, they can request more specific instructions by pressing an icon next to the text instructions, as shown in Fig. 1e. Each level of hints reveals more information about the strategy needed to progress the solution and suggests the use of specific command blocks. The results showed that the majority of students (52.88%) were unaware of the existence of this feature. A significant portion of students (30.77%) used the first level of hints, which is the default option. Only 16.35% of students discovered the option and used the second or third level of hints. The fifth question (Q5) of the questionnaire assessed the usefulness of the level completion information report. The game displays the total number of submitted code lines and prompts the students to retry their solution if it is not optimal by providing a message as shown in Fig. 2. The results showed that the majority of students either proceeded to the next level without retrying (76.92%) or did not understand or ignored the message (7.69%). Only a small proportion of students, 15.38%, chose to retry their solution with fewer blocks after viewing the prompt.

The second section of the questionnaire refers to the game characteristics. The sixth question (Q6) assessed students' reactions to the game theme. A majority of the students expressed their enjoyment of the theme, indicating they found it appealing (extremely 49.04%, very much 13.46%). On the other hand, one-third of the students found it indifferent or did not like it, (moderately 24.04%, little 6.73%, and not at all 6.73%). The seventh question (Q7) evaluated students' views on the game's graphics. Results indicated a mixed reaction, with half of the students appreciating and finding appealing the 2D perspective and pixel-art sprites (very much 20.19%, extremely 27.88%). Meanwhile, the remaining students expressed a less positive view about the graphics (moderately 21.15%, A little 19.23%, and Not at all 11.54%). Similarly, the results of the eighth question (Q8) about the musical aspect of the game, elicited mixed responses from the participants. Approximately half of the students reported a positive attitude towards the game's music, with 10.58% stating that they liked it very much and 30.77% stating that they liked it extremely. However, the remaining participants showed a less favorable reaction (moderately 19.23%, a little 19.23%, and not at all 28.85%). The last question (Q9) of the second section of the questionnaire aimed to estimate the students' perspective on the programming puzzles presented in the game. The game designed a range of puzzles that leveraged the elements of Minecraft, such as trees, rocks, and animals, to create engaging learning scenarios. A significant proportion of students reported that they enjoyed this aspect of the game (Very Much 12.50%, Extremely 30.77%) while the rest held a moderate to a lesser favorable response (Moderately 23.08%, A little 19.23%, Not at all 14.42%).

The last section of the questionnaire involves estimating the game's impact on the students. The game utilized a block-based interface to display commands and develop mini-programs. The tenth question (Q10) queried the students' enjoyment of programming in this type of environment. The majority of the respondents indicated positive feelings towards the experience (Extremely 32.69%, Very Much 14.42%) with the remaining expressing a moderate opinion or dislike (Moderately 25.00%, A little 18.27%, Not at all 9.62%). The last question (Q11) evaluated the students' willingness to participate in another Hour of Code game based on their

experience with Minecraft adventures. Most of the participants (71.15%) reported a positive response (28.85% negative).

As previously indicated, in the course of the experiment, the supervising teacher manually recorded the progress of the students in a spreadsheet. This resulted in the creation of a table that anonymously stored the total number of completed puzzles and the number of inefficient solutions in terms of the number of blocks used. This collection of data allows the analysis of the student's performance in relation to the support efficiency of the game. Table 6 and Figure 3 display the distribution and descriptive statistics of the maximum level reached. The quantification of student performance is achieved through a weighted percentage computation of efficient solutions, as shown in Eq. (1). The total number of efficient solutions for each student is determined by subtracting the inefficiently completed levels from the reached level. The result is weighted based on the maximum level of the game which is thirteen.

$$Performance = \frac{Completed\ levels - Inefficient\ solutions}{13} \times 100 \tag{1}$$

Table 6 The descriptive statistics of the collected data ($N = 104$)

	Minimum	Maximum	Mean	Std. Deviation
Puzzles completed	5	12	9.09	1.57
Insufficient solutions	0	9	4.36	1.96
Correct solutions	1	9	4.73	1.56
Performance	7.69	69.23	35.47	13.06

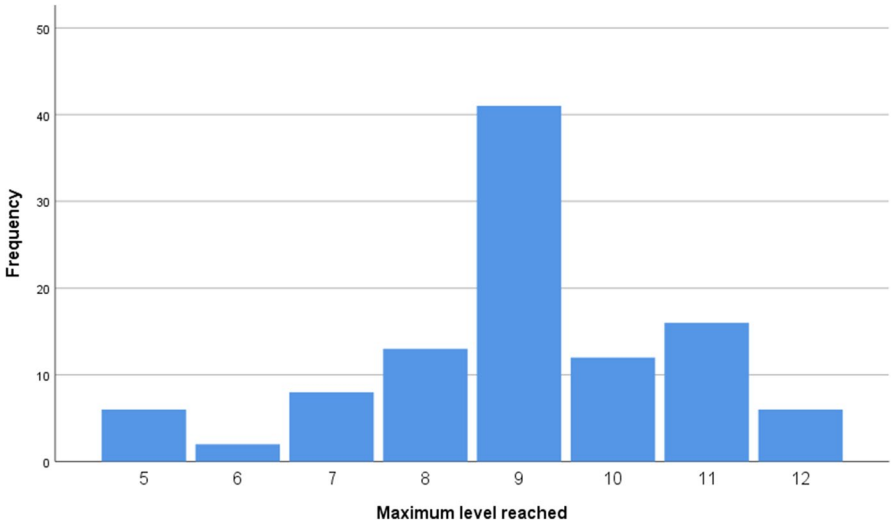

Fig. 3 The distribution of completed levels

Table 7 Spearman's test results for correlation between performance and questionnaire responses

ID	Question	r_s	n	p	Correlation strength with Performance
Q1	How helpful were the text instructions in solving the puzzles?	.674	104	.000	Positive Strong
Q2	How helpful were the video presentations in solving the puzzles?	−.116	104	.241	Not correlated
Q3	How helpful was the display of the number of blocks for the best solution in the workspace?	.574	104	.000	Positive Moderate
Q4	What instruction mode did you use most frequently?	.592	104	.000	Positive Moderate
Q5	What was your reaction when you were prompted that it is possible to complete the level with fewer lines of code?	.272	104	.005	Positive Weak

To investigate the impact of game support systems on student performance, a series of correlation tests were conducted. The responses of each student to questions Q1 to Q5 in the questionnaire were ranked ordered into numerical values where lower values indicate less effectiveness in received support. For instance, responses such as "I did not know that such an option existed" were ranked 1, whereas "A little" responses were assigned a rank of 2, and so forth. The Kolmogorov-Smirnov test was conducted to examine the normality of the data on the question (Q1-Q5) variables. Results concluded ($p = 0.00 < .01$) that data were not normally distributed in all variables. Therefore, the non-parametric test of Spearman's rank correlation was computed to assess the relationship between student performance and questionnaire results. The results indicate significant positive correlations ($p < .01$) between performance and all the questions except Q2 ($p = .241 > .01$) as shown in Table 7.

6 Discussion

HoCs Minecraft Adventurer is a SG and in this context, students are expected to engage in the learning process by applying their newly acquired knowledge, skills, and strategies to solve programming puzzles. To facilitate this process, it is imperative that the game provides effective and efficient support mechanisms that are easily understood and adequately presented. The results of the questionnaire reveal that a substantial portion of students did not find the support methods of HoC Minecraft Adventurer effective. We argue that many design choices involving the delivering techniques and UI need to reshape and adapt to student actions. This section will explore the possible reasons for failed support and suggest improvements and alternative methods for future implementations of HoC and similar SGs.

The Q1 findings show that the majority of students found the text instructions to be moderately or less helpful. Although text instructions are commonly used in text programming environments, this approach is less preferable on SGs. The expository text is often perceived as "dry and boring" by students reducing its learning efficiency (Ginns et al., 2013). Research has shown that interactive, visual, and

hands-on methods, such as working examples, can be more effective in promoting learning (Toukiloglou & Xinogalos, 2022). During the programming phase, students need to think critically, experiment, and apply what has been learned from previous problems. HoC relies on static predetermined tips to support students throughout the game, regardless of their progress, past mistakes, and knowledge level. On the contrary, static text can be replaced with adaptive support, which provides help that is tailored to the student's knowledge level. Adaptive tips and adaptive working examples have been shown to be more effective than static text as they provide feedback according to player learning models (Toukiloglou & Xinogalos, 2023).

Additionally, placement, content, and delivery of text instructions are critical elements of instructional design that can impact their effectiveness on the learning experience of the player. Well-designed text instructions can enhance learning and performance, while poorly designed instructions can hinder progress and even cause frustration (White, 2014). During the puzzle-solving process, the student's attention is primarily focused on the game world where the puzzle is presented, and the workspace where the program is constructed. The screen real estate of text tips is small compared to the other game elements and can make them easily overlooked. Following the widely accepted System Usability Scale (SUS) requirements, it seems that more attention should be paid on ensuring that the various functions of the game, including the text tips in our case, are well integrated (Bangor et al., 2008). Moreover, the color of the text in the tips is white on a gray background (Fig. 1e) which makes it blend in instead of standing out with a vivid visual representation. As noted by Petri et al. (2016), the text font and colors must be well blended and consistent, while the fonts both in terms of their style and size must be easy to read. This observation can be combined with the results of Q4 which concluded that most of the students did not use, understand, or know about the scaffolding tip strategy. The tip icon is small in size and the number of available tips indicator can be easily missed. This results in a significant part of the support method being omitted and violates the widely accepted Nielsen's usability heuristic regarding the visibility of system status (Nielsen, 1994). Moreover, the number of available tips, content, and order of appearance can be improved. The game combines instructions for the puzzle solution with story elements in the same sentence which can be misleading as it does not always translate into actionable strategies. Instead, the game should provide a simple and natural dialogue that does not contain information that is not relevant, while it should be *"easy to discriminate action alternatives"* (Nielsen, 1994; p.3). Providing guidance and affordance, as well as clear and relevant feedback are considered important features for games in terms of their pedagogical usability (Sanchez, 2011). The scaffolding tips provided may also not be in line with the increasing difficulty of the puzzles, leading to confusion and frustration. A solution to this issue could be to provide fewer tips for the easier puzzles, and more in-depth, layered tips for the more challenging puzzles.

Another method of support of the game is the video presentation where usually a new programming concept or mechanic is introduced. Although HoC videos can be characterized as entertaining and informative, most of the students found them not helpful. This was probably due to the fact that the video's main purpose was to inspire and engage students rather than help them solve a puzzle. Adding

entertaining video clips to a multimedia lesson can distract the learner, reducing their ability to engage in deeper cognitive processing and build meaningful learning outcomes (Mayer et al., 2020). Therefore, the instructional video should prioritize building knowledge rather than promoting excitement. The current form of videos in HoC might be a missed opportunity to enhance the learning experience since students in online courses benefit from visually seeing instructors on-screen (Belt & Lowenthal, 2021). We suggest changing the content and utilizing it as a support method to present a working example or as a tool to promote the story. Another factor that might have contributed to the negative results is the practicality of the video presentation in a school computer lab environment. Videos can be less effective in a group setting where each student needs personal headphones to listen to the audio without disrupting other classmates. This requirement was not fulfilled during the experiment and decreased the efficacy of the video as a support method. Likewise, the absence of personal headphones resulted in a diminished positive impact of music as a game element and a less favorable response in the answers for Q8. The group setting hindered the personal experience with music, leading to difficulty in concentration and negatively affecting the overall experience. As a consequence, most of the students turned off or drastically reduced the sound resulting in a loss of immersion in this aspect of the game. Additionally, the shared sound disrupted students who preferred to concentrate in silence.

The Hour of Code (HoC) games are created to be self-sufficient and allow students to learn coding independently, without teacher supervision. This claim is advertised on the official website where it states that anyone can try a one-hour tutorial that is designed for all ages (Hour of Code Activities, n.d.). During our experiment, students were left to discover the various support systems and game mechanics on their own. However, we found that many of the game's UI elements were unnoticed or poorly understood by the students, as previously observed in Q1 and Q4 regarding the text instructions. This is also evident in Q3 where students were divided on the helpfulness of the number of blocks indicator. Although the indicator can guide students toward the correct solution by showing the number of blocks required, students may not be aware of its existence or purpose. The game did not provide any information on the purpose of the indicator. Without proper presentation, students may not know how to use this indicator effectively to improve their programming skills. The size, placement, or design of the indicator could also impact its usability and effectiveness.

The results of Q5 showed that a mere 15% of students attempted to rewrite their solution when prompted to do so in order to complete the level with fewer lines of code. Several factors can contribute to the reasons for that behavior such as boredom, lack of understanding that it is an option, and failure to read the message. Another possible explanation is that a non-optimal solution is non-mandatory for the game to continue. This aspect has some severe implications, as it is possible for students to continue solving puzzles without utilizing proper programming structures. For instance, instead of using the iteration block, students may submit a sequential solution with equal results but insufficient code quality. As a consequence, students may progress to the next game level without a proper understanding of essential programming structures, since the game did not

prevent them from doing so. The data collected by the teacher administrating the session concluded that most students submitted less efficient solutions (Table 6). This implies that students' focus shifted from code learning to level advancement. This game mechanic resulted in students progressing without a proper understanding of essential structures. We argue the need to make efficient solutions mandatory or clearly indicate insufficient programs to promote proper learning.

The selection of Minecraft as the theme for the SG was supported by the fact that it is a hugely popular game, especially with younger audiences (Minecraft Player Count and Stats, 2023). As such, most of the students in our study were already familiar with the game world and lore, which is reflected in the overwhelmingly positive results of Q6. By using Minecraft as the game theme, students were able to better understand the rules that govern the game world and follow the storyline. This allowed the game to skip detailed tutorials since users intuitively knew how to interact with game objects such as trees can be chopped to create wood palettes, rocks can be mined to craft iron tools, walking on water or lava is dangerous, etc. However, 13.4% of students who did not like the game theme were likely unfamiliar with Minecraft or simply did not like the setting. Using a commercial game as the basis for a serious game carries some risks as it may result in conflicting design goals and lead to compromised learning outcomes. Commercial games are not typically designed with pedagogical objectives in mind which can make it difficult to repurpose them for educational purposes (Reyes-de-Cózar et al., 2022). Nonetheless, as the current study data suggest in the case of Minecraft Adventurer, with careful consideration and modification commercial games can serve as a valuable foundation for the development of effective SGs. However, the game should include support systems for users who are not familiar with the franchise, which could be made skippable to allow users to choose according to their preferences.

The student's opinions on the game graphics (Q7) were evenly split. While the game accurately represents the Minecraft world and its iconic block shapes and textures, it differs from the original game as it is presented in a top-down 2D format rather than an open-world 3D format. The HoC Minecraft Adventurer game runs on a webpage and uses Javascript, making it accessible to a wide range of computers with low specifications. The game's design aims to promote the HoC movement, even in countries with limited resources. Regardless, the most popular commercial games typically have 3D graphics and higher hardware requirements (Wikipedia, 2023), which makes HoC Minecraft look outdated in comparison. To address this, it could be beneficial to offer two versions of the game, one in 3D and another in 2D, to allow players to choose the version that best suits their system although this would increase cost production.

The HoC Minecraft Adventurer game comprises 13 puzzles that are intended to be solved within a single class session. These puzzles are specifically designed to teach essential programming structures, such as sequence, iteration, and conditions, which are typically incorporated in introductory programming SGs. However, the game's approach of fitting all three structures into one class session results in a steep learning curve for novice students. Difficulty increases disproportionately as the game progresses and students do not have adequate time and experience to process

the knowledge related to the new programming structures. The mixed results on Q9 about the student opinion on programming puzzles are possibly due to this reason, although each individual puzzle as a standalone construct is well designed. Students that struggle to continue the game, especially after level 9 (Figure 3), have a negative first contact with programming, which is the opposite effect of what HoC intends. We suggest a decrease in the number of educational objectives, such as the condition, to allow more time spent on others and give students enough time to grasp the new knowledge. Another solution is to increase the number of puzzles and split the HoC experience into two or more sessions.

Results from Q10 indicate that the majority of students favor the block-based environment. This programming style has proven effective for teaching programming to beginners as it is designed to target this group (AbdulSamad & Romli, 2022). The block-based programming languages typically include visual representations of code which can help learners better understand programming concepts. They allow users to avoid the syntax errors that commonly occur when writing code in text form and focus more on logical thinking. In recent years, the extensive adoption of this style led to its inclusion in most programming environments, including SGs. As a result, many students are already familiar with this programming style, which reduces the period of adjustment required when interacting with a new application.

According to the results displayed in Fig. 3 and Table 6, students encountered difficulties in terms of puzzle completion and knowledge acquisition. The Mean success rate in students that is calculated by the percentage of correct submissions over the total number of completed puzzles is very low (Mean = 53.06, SD = 17.87, N = 104). This indicates that almost half of the submitted solutions were inefficient and students were moving to the next level without fully understanding the programming concepts the game was designed to teach them. Additionally, the mean of completed puzzles (Mean = 9.09, SD = 1.57, N = 104) indicates that most of the students did not reach the final levels of the game or gave up when the puzzles became too complex. Students that did not fully understand the programming structures normally should have a hard time completing levels that require a deep understanding of the structures and how to combine them. Also, the maximum level reached was 12 out of 13, which means that for this group of novice students, an hour (45′) of code was not enough time to complete the game. Those results are direct evidence of the problems described above for the support system of the game.

The correlation analysis (Table 7) reveals a meaningful relationship between students' utilization of the support system and their overall performance scores. Particularly, a strong positive correlation was observed between performance and text instructions (Q1), suggesting that students who engaged with the instructions submitted more efficient puzzle solutions. Conversely, the impact of video presentations (Q2) on performance was found to be negligible, as most students did not watch them or the content did not contribute significantly to the puzzle outcome. The number of blocks for the best solution indicator showed a moderate positive correlation with performance, implying that most of the students who understood its purpose incorporated the information into their solutions. Similarly, a moderate positive correlation emerged between performance and instruction mode,

Table 8 The most frequently reported student comments

Student comment
It was very hard after level 5
I got stuck at level 8
It was a nice experience
I liked the game a lot and it made me like Minecraft even more
It was a nice game but I wish it had better graphics
I did not like that in some solutions I had to create very long programs
I found the puzzles difficult but I managed to progress through the levels
I liked the game. However the difficulty increases rapidly and I didn't have any help
The game was good but most of the puzzles were too hard to solve

revealing that students who effectively utilized more than one level of help, benefited from it to a moderate extent. Finally, the message prompting students that a better solution is possible after their initial submission demonstrated a weak correlation with performance. Even though some students comprehended the message and attempted to improve their solutions, it did not consistently result in better scores for their subsequent attempts. Overall, the results highlight the importance of effective support system usage in enhancing students' puzzle-solving capabilities. If more students understood and utilized the support systems in the manner intended by the designers, the performance would have been notably improved.

Lastly, the observations made earlier are supported by the comments provided by the students. In the questionnaire, a section was included to allow students to freely express their thoughts and experiences during the HoC session. The comments provided by the students are presented in Table 8, and they are primarily written in informal language.

In spite of the mixed results regarding the support systems and UI deficiencies, the game has been found to be effective in engaging students, as evidenced by the results of Q11. Over two-thirds of the students expressed their willingness to try another HoC game. Furthermore, there are currently three additional HoC Minecraft-themed games, as well as other games with similar designs, available for students to explore and enhance their programming skills.

7 Limitations

It is important to note that, due to the nature of the study, the sample may not be representative of the population of all elementary school students who are novices in programming. Therefore, the results of the study should be interpreted with caution, and generalization to other populations should be done with care. Another limitation of the study was the absence of headphones, which compelled students to lower or turn off the sound to avoid disrupting their classmates in the computer lab. This restriction may have influenced the results of Q9, as noted in

the discussion section. Lastly, one of the limitations of this study is that detailed data on the performance of individual students at each level of the game was not collected. Instead, only the total number of inefficient solutions and the final level reached per student were recorded due to the challenges of managing the class with only one administrator. As a result, it was not possible to identify the specific levels at which students faced the most challenges, which could have enabled more targeted efforts to improve support for those areas.

8 Conclusion

The present study investigated the effectiveness of the support systems and UI of the SG Minecraft Adventurer, which serves as a typical example of a HoC game for learning programming. Both qualitative and quantitative methods were employed to assess the impact of the game's design on the learning outcomes of the users. The findings of this study indicate that a considerable proportion of the students did not find the support methods and UI of the HoC Minecraft Adventurer to be effective (RQ1). The correlation analysis conducted to examine the relationship between student performance and the utilization of support systems revealed that the support system and UI implementation affect student game performance (RQ2). Specifically, students who either did not use the support system or employed it incorrectly due to a lack of comprehension exhibited a reduction in their performance in solving the puzzles, consequently diminishing effectiveness in learning outcomes. Furthermore, the study recommends various improvements to the UI and support mechanisms (RQ3). Those modifications can be applied not only to the eleven compatible HoC games but also to other SGs that follow similar design choices. As HoC games are widely used by millions of students and teachers worldwide, we hope that these suggestions will be considered to enhance an already successful learning platform.

Funding Open access funding provided by HEAL-Link Greece.

Data availability The authors declare that the data supporting the findings of this study are available within the article.

Declarations

Conflict of interest None.

References

Abdellatif, A. J., McCollum, B., & McMullan, P. (2018). Serious games: Quality characteristics evaluation framework and case study. In 2018 IEEE Integrated STEM Education Conference (ISEC) (pp. 112–119). IEEE.

AbdulSamad, U., & Romli, R. (2022). A comparison of block-based programming platforms for learning programming and creating simple application. In F. Saeed, F. Mohammed, & F. Ghaleb (Eds.), *Advances on Intelligent Informatics and Computing* (Vol. 127, pp. 630–640). Springer International Publishing. https://doi.org/10.1007/978-3-030-98741-1_52

Apple. (2022). Human Interface Guidelines. Apple Developer. Retrieved March 29, 2023 from https://developer.apple.com/design/human-interface-guidelines/

Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction, 24*(6), 574–594.

Basawapatna, A., Repenning, A., & Savignano, M. (2019). The zones of proximal flow tutorial: Designing computational thinking Cliffhangers. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 428–434. https://doi.org/10.1145/3287324.3287361

Belt, E. S., & Lowenthal, P. R. (2021). Video use in online and blended courses: A qualitative synthesis. *Distance Education, 42*(3), 410–440. https://doi.org/10.1080/01587919.2021.1954882

Buwalda, M., Jeuring, J., & Naus, N. (2018). Use expert knowledge instead of data: Generating hints for hour of code exercises. *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, 1–4. https://doi.org/10.1145/3231644.3231690

Code.org 2021 Annual Report. (n.d.). Code.org. Retrieved March 29, 2023 from https://code.org/about/2021

DIS, I. (2009). 9241–210: 2010. Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems. International Standardization Organization (ISO).

Effenberger, T. (2019). Towards adaptive hour of code. In S. Isotani, E. Millán, A. Ogan, P. Hastings, B. McLaren, & R. Luckin (Eds.), *Artificial Intelligence in Education* (pp. 339–343). Springer International Publishing. https://doi.org/10.1007/978-3-030-23207-8_62

Effenberger, T., & Pelánek, R. (2018). Towards making block-based programming activities adaptive. *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, 1–4. https://doi.org/10.1145/3231644.3231670

GhasemAghaei, R., Arya, A., & Biddle, R. (2016). Evaluating software for affective education: A case study of the affective walkthrough. In C. Stephanidis (Ed.), *HCI international 2016 – Posters' extended abstracts* (Vol. 618, pp. 226–231). Springer International Publishing. https://doi.org/10.1007/978-3-319-40542-1_36

Ginns, P., Martin, A. J., & Marsh, H. W. (2013). Designing instructional text in a conversational style: A Meta-analysis. *Educational Psychology Review, 25*(4), 445–472. https://doi.org/10.1007/s10648-013-9228-0

Hour of Code Activities (n.d.). Hour of Code. Retrieved March 29, 2023 from https://hourofcode.com/gr/gb/learn

Johnson, J. (2020). *Designing with the mind in mind: Simple guide to understanding user Interface design guidelines* (3rd ed.). Morgan Kaufmann.

Lanyi, S., Brown, J., Standen, P., et al. (2012). Results of user interface evaluation of serious games for students with intellectual disability. *Acta Polytechnica Hungarica, 9*, 225–245.

Martinez, K., Menéndez-Menéndez, M. I., & Bustillo, A. (2022). A new measure for serious games evaluation: Gaming educational balanced (GEB) model. *Applied Sciences, 12*(22), 11757.

Mayer, R. E., Fiorella, L., & Stull, A. (2020). Five ways to increase the effectiveness of instructional video. *Educational Technology Research and Development, 68*(3), 837–852. https://doi.org/10.1007/s11423-020-09749-6

Mikovec, Z., Salvik, P. and Zara, J., (2009). Cultural heritage, user interfaces and serious games at CTU prague. *15th International Conference on Virtual Systems and Multimedia*, pp. 211–116.

Minecraft Player Count and Stats (2023). Video Game Statistics. Retrieved March 29 from https://videogamesstats.com/minecraft-statistics-facts/

Moizer, J., Lean, J., Dell'Aquila, E., Walsh, P., Keary, A. A., O'Byrne, D., ... & Sica, L. S. (2019). An approach to evaluating the user experience of serious games. *Computers & Education, 136*, 141–151.

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 152–158).

Petri, G., von Wangenheim, C., & Borgatto, A. (2016). MEEGA+: An evolution of a model for the evaluation of educational games. Brazilian Institute for Digital Convergence. *Technical Report, 3*, 1–40.

Piech, C., Sahami, M., Huang, J., & Guibas, L. (2015). Autonomously Generating Hints by Inferring Problem Solving Policies. *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, 195–204. https://doi.org/10.1145/2724660.2724668

Reyes-de-Cózar, S., Ramírez-Moreno, C., & Barroso-Tristán, J. M. (2022). A qualitative analysis of the educational value of commercial video games. *Education Sciences, 12*(9), 584. https://doi.org/10.3390/educsci12090584

Roboblocky (n.d.). Roboblocky: The ultimate coding platform for kids. Retrieved August 9, 2023, from https://roboblocky.com/

Sanchez, E. (2011). Key criteria for game design. A framework. In N. Baldissin, S. Bettiol, S. Magrin, & F. Nonino (Eds.), *Business game-based learning in management education* (pp. 79–95)

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, 60*(7), 1615–1636.

Toukiloglou, P., & Xinogalos, S. (2023). Adaptive support with working examples in serious games about programming. *Journal of Educational Computing Research, 61*(4), 766–789.

Tynker (n.d.). Coding for kids, kids online coding classes & games. Retrieved August 9, 2023, from https://www.tynker.com/

White, M. M. (2014). *Learn to play: Designing tutorials for video games*. CRC Press.

Wikipedia. (2023). List of best-selling video games. In Wikipedia, The Free Encyclopedia. Retrieved March 19, 2023, from https://en.wikipedia.org/w/index.php?title=List_of_best-selling_video_games&oldid=1012864980

Yauney, J., Bartholomew, S. R., & Rich, P. (2021). A systematic review of "Hour of Code" research. *Computer Science Education*, 1–33. https://doi.org/10.1080/08993408.2021.2022362