Design and Analysis of a Virtual Reality Game to Address Issues in Introductory Programming Learning

Chyanna Wee(✉) and Kian Meng Yap

Sunway University, 47500 Subang Jaya, Selangor, Malaysia
15060197@imail.sunway.edu.my, kmyap@sunway.edu.my

Abstract. The field of computer science has not shied away from employing game-based learning and virtual reality techniques for computer programming education. While a plethora of game-based, virtual reality or combinations of both solutions exist, most are developed as an alternative to traditional lessons where students focus on learning programming concepts or languages. However, these solutions do not cater to problems students face when learning programming that is mainly caused by the abstract nature of programming, misconceptions of programming concepts and lack of learning motivation. Hence, in this paper, a framework to address the abstract nature of programming, common programming misconceptions and motivational issues is developed. The framework consists of three modules that correspond to each issue powered by a simulation engine. To address the abstract nature of programming, programming concepts will be represented with concrete objects in the virtual environment. Furthermore, to address common programming misconceptions, simulation techniques such as interactions and player perspective will be utilised. Lastly, motivational game elements will be employed into the simulation to engage students when learning through the system. Results gathered from questionnaires indicated that users were generally satisfied with the virtual experience developed from the framework.

Keywords: Computer science education · Educational games · Virtual reality

1 Introduction

By the year 2024, it is predicted that the availability of computing-related jobs would increase by twelve-point five percent [1]. This fact is also reflected in the enrolment rates for computing-based courses where student admissions have increased by seven percent from the year 2017 to 2019 [2]. However, recent reports show that the dropout rates for computing courses is at nine-point eight percent which is the highest between other Science, Technology, Engineering and Mathematics (STEM) based courses [3].

This is an issue that has concerned many researchers and has sparked and initiated various studies to determine reasons for the high non-continuation rates. The reasons commonly attributed to student dropouts have ranged from substandard teaching,

N. Shaghaghi et al. (Eds.): INTETAIN 2020, LNICST 377, pp. 243–254, 2021.
https://doi.org/10.1007/978-3-030-76426-5_16

negative experiences and low sense of belonging [4]. Furthermore, students who have un-realistic expectations prior to joining a computing course can also contribute to the decrease in retention rates because often, they fail to fulfil these expectations [5]. Consequently, researchers like Tan, Ting and Ling [6] and Medeiros [7] have taken a different approach and went on to investigate dropout factors from a learning perspective instead. More specifically, these studies analysed issues that students face when learning computer programming. Some of the main issues include the lack of learning motivation, the abstract nature of computer programming concepts and misconceptions of programming concepts.

Due to the use of high-level programming languages that are designed to provide a certain degree of abstraction from the details of the computer to the user, students often find it difficult to understand programming concepts. Particularly, students find it hard to relate programming concepts to real-life and how learning these concepts can solve real world problems [8]. Due to this, students may feel unmotivated to continue and participate in programming lessons. Moreover, students are also more prone to developing misconceptions if they do not fully understand the programming concepts introduced to them. This in turn, can cause unwarranted syntax errors when students attempt to develop a program which further decreases motivation and ultimately leads to student dropouts [9].

Over the years, researchers have come up with different solutions to make computer programming lessons more engaging. For instance, the Game-Based Learning (GBL) technique is commonly utilised where games are incorporated into the learning process. This makes for a highly engaging way to encourage learning and has been proven to lead to knowledge acquisition [10]. Other than that, virtual reality (VR) simulations are also prominent for developing immersive lessons for the whole classroom. More recently, researchers have also employed both techniques to develop solutions that maximise the advantages of both GBL and VR.

However, existing GBL and VR applications are mainly focused on introducing programming concepts and programming syntax. While this is good for students to acquire the "gist" of programming, it does not help students understand how abstract concepts like variables, lists and arrays works. Particularly, how these data structures are stored and accessed in the computer memory. To further aid in the understanding of abstract data structures, visualization and analogies are also necessary. This is important to offload cognitive loads to better incorporate programming concepts into the mental models of students [11]. Other than that, existing applications also do not address common misconceptions to students. This is important as unaddressed misconceptions may lead to problems as the student progresses through the course. Hence, this paper aims to reduce the abstractness of programming concepts and address misconceptions in a motivational and engaging manner.

2 Related Works

This section aims to highlight past works that are closely related to the system. More specifically, applications that employ both GBL and VR will be reviewed. Along with this, research gaps will be identified.

2.1 Virtual Reality and Game-Based Learning

Related works presented in this section will feature VR applications developed to aid in learning computer programming that incorporates game elements. In this context, game elements refer to objectives that are presented to users. This can range from making users complete challenges such as getting from one point to another for the sake of progressing through the experience.

For instance, Vincur et al. [12] developed a VR experience called "Cubely". Cubely employs block-based programming in hopes to make learning programming seem more approachable to new learners. The system features interactable cube blocks that can be used to build small programs. These blocks are labelled with typical programming statements such as for, if and else. The programs built with these blocks will then act as commands to control an animated character in overcoming various challenges. To test the feasibility of the system, a total of nineteen participants were recruited. The system garnered positive feedback in terms of user friendliness and its immersive quality compared to online code camps.

To aid in the learning of Object Oriented Programming (OOP) concepts, Bouali et al. [13] developed a VR game known as "Imikode". The basis of the system is to allow users to program and observe how a specific set of code affects the virtual world. The system allows learners to visualize concepts such as object instantiation, method calls and get/set commands in the virtual environment. For instance, a fox appears in the virtual environment when an object instantiation of a fox is made. The system also features a character that issues out challenges and helps users throughout the experience. As of now, there are no tests done yet to determine the feasibility of the system.

Similarly, Chen et al. [14] also developed a system that employed VR and GBL techniques to help students learn programming. Essentially, the system encourages programming learning in two ways. Firstly, the system allows users to utilize code in creating obstacles for game levels in the virtual environment. Secondly, users can also progress through the game by completing programming-based tasks. This means that users that are tasked with the role of the level designer will first employ "enemy robots" in the virtual scene with commands that bares similarity to the Java programming language. Users who then go through the experience must progress through the game by acquiring hints and answering programming questions while overcoming obstructions that is set up by the level designer. To determine the feasibility of the system, the authors organized a bootcamp where the application is showcased. Results showed that out of fifty-three participants of age nine to thirteen, fifty-two participants found that the system was engaging and promoted learning effectiveness.

Segura et al. [15] also utilized block-based programming in developing a VR game called "VR-OCKS". The game requires users to complete puzzles by controlling an ingame character. Users progress through the game by building programs that acts as commands to get from one point to another while avoiding obstacles. The commands are built with action blocks that represents basic programming concepts and such as for loops, while loops and conditional statements. There are also blocks that represents simple actions such as turn and move forward. Essentially, users place and combine various blocks in the virtual environment to control the character. This allows users to witness how the program that they have built works, fostering programming skills. To test

the feasibility of the system, a total of forty participants were recruited. The participants are then broken up where twenty of them have used the system while the other twenty acted as the control group. Each group were then further divided to try out the "Kodu" and "Blockly" block-based programming applications. The results from the experiment showed that those who had experience with VR-OCKS completed twenty five percent more levels than those who didn't.

2.2 Research Gaps

There are three apparent research gaps that can be determined from literature reviewed in the previous section. Firstly, all existing VR applications employing GBL are mainly focused on introducing programming concepts with a heavy emphasis on syntax. This means that there is a paucity of solutions when it comes to addressing common issues faced by students when learning introductory programming. As established, the abstract nature of programming can make understanding concepts a gruelling task if one is unable to form correct mental models. This can cause further issues such as misconceptions and loss of learning motivation. Hence, a solution that addresses these issues are necessary to enhance understanding of programming concepts beyond just learning the syntax of a particular language.

3 Methodology

In order to reduce the abstractness of programming concepts and address misconceptions related to programming in an engaging manner, we propose a virtual reality game. The system mainly consists of three modules that is built with Unity 3D for the Oculus Rift. Figure 1 shows a general framework of the system which consists of the abstract module followed by the misconception module and the motivational module. Sects. 3.2 to 3.3 explains each module in better detail. However, considerations and reasons for developing the system in a certain way is first discussed in Sect. 3.1.

3.1 System Considerations

Like any other system, some choices are made during the development process. These ranges from the type of interactions to employ, the medium of instruction and the simulation perspective. Figure 2 shows these in more detail. Firstly, there are predominantly two type of interactions present in virtual reality simulations. Some systems employ direct interactions with natural hand gestures such as grabbing and pointing while others only allow indirect interaction through a user interface. Secondly, systems developed for use in computing education can deliver interactive lessons with either a text-based or block-based medium. The default medium for teaching programming is text-based. The most popular system that employs block-based programming is Scratch where users build small programs with command blocks. Thirdly, virtual simulations can be developed in the first perspective where the user control themselves and the third perspective where the user controls another character.

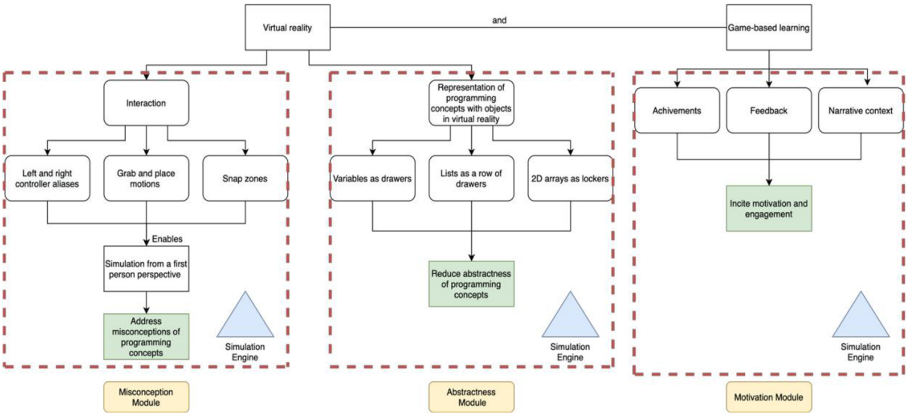

Fig. 1. Framework of proposed methodology.

According to Norris et al. [16], direct interactions with the environment can improve knowledge acquisition. Consequently, Nederveen et al. [17] acknowledged that learning through a first-person perspective can also improve knowledge acquisition. Hence, to improve knowledge acquisition the system will employ direct interactions and will be developed in the first-person perspective. To avoid the need to re-learn syntax, all tasks presented to the user will be delivered with the text-based medium of instruction.

The system is developed with Unity 3D for the Oculus Rift. Frameworks such as the Oculus Integration Package and the Virtual Reality Toolkit (VRTK) are used to implement elements such as teleportation (locomotion), hand presence, interaction and audio. Both frameworks provide useful scripts that speeds up the development process.

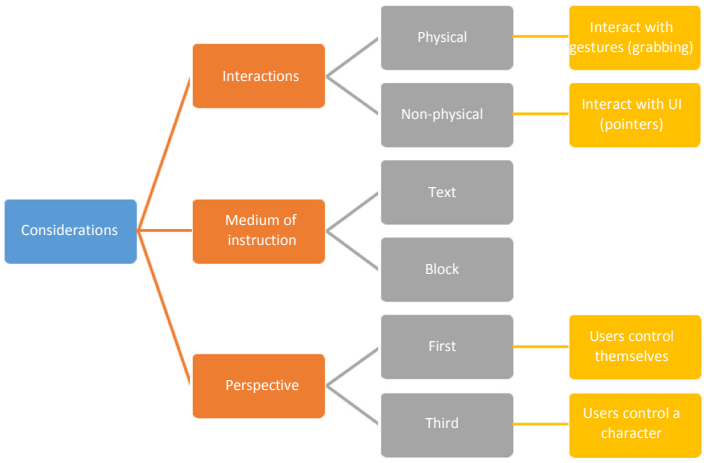

Fig. 2. System considerations.

3.2 Abstract Nature of Programming Concepts

As seen in Figure 1, concepts such as variables, lists and arrays are represented in the virtual environment with real-life objects to aid in the understanding of abstract programming concepts. To determine suitable objects that can be used to represent each concept, attributes such as the ability to label (to mirror how variable, list and array names are used to "label" associated data) and store items (to mirror how values are assigned to a variable, list and array) are taken into consideration. While objects like buckets, envelopes and honeycombs are considered, drawers and lockers are picked instead because it is more conventional to label and store objects in drawers and lockers as opposed to buckets, envelopes and honeycombs. Figure 3 shows how the concepts are represented in the virtual environment where variables, lists and 2D arrays are represented with drawers, a row of drawers and lockers consecutively. This allow students to form concrete analogies to better understand the abstract concepts.

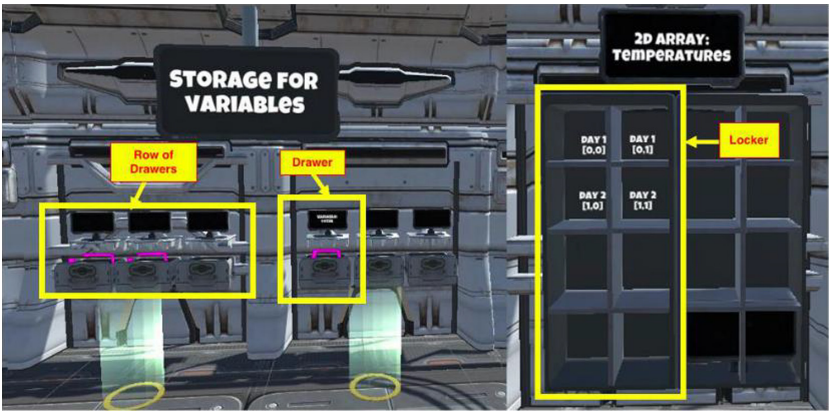

Fig. 3. Variables, lists and 2D arrays represented with drawers, row of drawers and lockers respectively.

To show how data is stored and accessed in the computer memory, the system first presents students with a line of code. This will act as the "challenge" students must complete to unlock new challenges and levels. Particularly, students will be presented with an assignment statement which consists of a data structure and a value. Students are then required to grab the correct data blocks and place it into the correct drawer. For example, if the challenge presented to the user is a variable declaration to assign the value 10 to a variable named total, students are required to grab the data block labelled with the value "10" and place it into the drawer labelled with the variable "total". Figure 4 shows the actions taken to successfully complete a challenge.

3.3 Misconceptions of Programming Concepts

As seen in Figure 1, the ability to interact with objects in the virtual environment in the first-person perspective largely enables the misconception module to address common

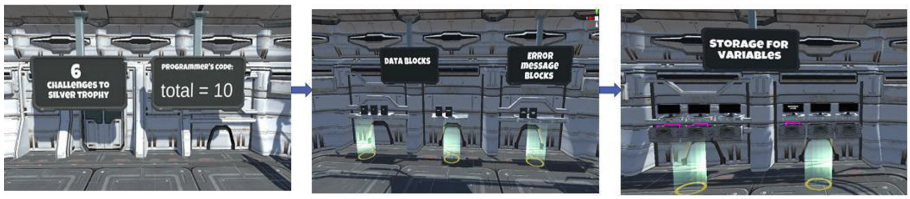

Fig. 4. Steps taken to complete a challenge: (1) Assignment statement presented as the "challenge", (2) Data blocks represent values, (3) Drawer representing the variable with purple snap zones that act as visual aids. (Color figure online)

misconceptions. Firstly, to allow students to feel a real sense of presence, left and right-hand aliases that resemble real-life hands are employed into the system. With the use of the default Oculus Rift touch controllers, natural hand gestures in the virtual environment such as grabbing and placing objects are possible. Consequently, snap zones provide visual interaction cues to further facilitate interaction. This enables students to efficiently carry out the tasks presented to them.

For the case of addressing programming misconceptions, the necessary actions taken to complete the challenge is similar to the one described in the previous section. However, instead of just placing data blocks into corresponding drawers or lockers, students must first identify whether the code presented to them possess any syntax errors. If so, students must identify the mistake and place the corresponding "error message" block onto the output station. For example, if the code presented to the user is 5 = count, students must grab the error message block labelled with "wrong assignment" and place it onto the output station (Fig. 5). Since this is a first-person simulation, students who initially could not identify why the code given is incorrect would eventually realise their mistake. For instance, if the student attempts to place the data block labelled with "count" into a drawer labelled with 5, the system simply does not allow this to happen. Consequently, by enforcing the action of placing the data block into the drawer, students can more easily comprehend why the structure of an assignment statement is declared a certain way.

Fig. 5. Misconception example: (1) Code presented to user is syntactically incorrect, (2) Error message blocks, (3) Output station where error message blocks are placed.

3.4 Motivation

As seen in Fig. 1, game elements such as a narrative context, instantaneous feedback and achievements are employed into the system to incite motivation. Firstly, a narrative context is introduced to students before they embark on any challenges in the tutorial scene. Students will be informed that they are currently in the computer's memory where data is stored. Then students would be told that their job is to act as the computer's "assistant" to manage incoming code that is entered by a user. Particularly, students are required to manage how data is stored and accessed in the computer's memory. This gives student's a sense of purpose, making it easier for students to participate in the simulation.

Secondly, instantaneous feedback is present in the system in the form of audio cues. Quite simply, the actions taken by the student is guided through both positive and negative audio ques. While the negative audio cues are used to indicate mistakes made, positive cues are employed to provide a confidence boost which can increase motivation to finish off tasks presented by the system.

Lastly, the system also features an achievement system that rewards students with virtual "trophies" after successful completion of challenges for a particular concept (Fig. 6). This acts as an indication of progress, as well as to foster user engagement.

Fig. 6. Trophies that can be earned during the course of the simulation.

4 System Evaluation

The system was evaluated with the aid of 13 participants between ages 17 to 20, of which 11 were female and 2 were male. 6 of the participants have had taken general computer science courses and were quite knowledgeable in some programming concepts albeit having no actual programming experience. On the other hand, 7 of the participants have never taken any computing-based courses. Hence, they were first briefed on some of the concepts that will be introduced in the virtual experience such as variables, lists and 2D arrays.

The participants were then asked to complete a series of tasks in the virtual environment with the Oculus Rift device equipped with a workstation that is rigged with a NVIDIA GeForce GTX 1080 GPU. After completing all the tasks, questionnaires were given out to assess and evaluate the system. The questionnaire mainly consists of questions that aims to judge user satisfaction and perceived outcomes of the developed system. Table 1 and Table 2 shows a summary of responses gathered from participants that were based on the Likert Scale.

Table 1. Summary of responses from satisfaction survey.

	1 (Strongly dissatisfied)	2 (Dissatisfied)	3 (Neutral)	4 (Satisfied)	5 (Strongly satisfied)	Mean
Representation of variables as drawers in aiding understanding				8	5	4.38
Representation of lists as a row of drawers in aiding understanding				7	6	4.46
Representation of 2D arrays as lockers in aiding understanding				6	7	4.54
Interactions with virtual objects in aiding misconceptions				4	9	4.69
Presence of audio cues in promoting motivation		1	2	4	6	4.15
Presence of achievements in promoting motivation		1	1	4	7	4.31
Presence of narrative role in promoting motivation			4	9		4.69

Table 2. Summary of responses from perceived outcome survey.

	1 (Strongly disagree)	2 (Disagree)	3 (Neutral)	4 (Agree)	5 (Strongly agree)	Mean
I now feel more motivated when learning programming			3	6	4	4.08
I prefer to have analogies presented to me visually in the virtual environment then spoken about in class verbally			2	3	8	4.46
The simulation is a good supplement to lessons taught in a traditional classroom				3	10	4.77
I prefer learning programming through games			1	3	9	4.62
I prefer learning programming through virtual reality experiences				6	7	4.54
I now feel more confident about my programming skills after the experience		1	3	4	5	4.00

5 Discussion

For analysis of the data gathered from the post survey, the average mean is calculated for every question. As seen in Table 1, participants were generally satisfied with the representation of variables, lists and 2D arrays as concrete objects in accordance to aiding their understanding of the concepts. This is mainly due to the fact that common objects such as drawers and lockers were easier to comprehend and the possibility of interacting

with the objects made previously abstract concepts more concrete. Participants were also satisfied with the way common misconceptions were simulated and addressed in the virtual experience. For instance, when presented with a syntactically wrong assignment statement, the nature of the simulation requires that the mistake is identified and corrected. This is done with assigning error message blocks to the output station as seen in Fig. 5. Lastly, while satisfaction scores for the presence of a narrative context, trophies and audio cues in promoting motivation were generally satisfactory, some participants expressed disaffection in terms of not getting celebrative cues such as confetti to make trophy collecting a more joyous experience. In terms of audio cues, some felt that the sounds were not loud enough.

Most participants agree that they feel more motivated to continue learning programming and thought that the simulation was a good complement with traditional classes. While most of the responses were positive, some may still prefer traditional lessons over unconventional mediums such as games. This may largely depend on whether the individual is an auditory or visual learner. Most participants also seem keener to learn with virtual reality experiences with some describing the experience as fun and new. In terms of perceived confidence levels, participants who had never taken any computer-based courses were more reluctant when it comes to feeling more confident after the experience. This may be attributed to a lower sense of belonging since they did not have prior knowledge on these concepts compared to those who have taken computing courses.

6 Conclusion and Future Work

This paper outlined several issues related to learning introductory programming. Firstly, is the need to address the abstractness of programming concepts where a framework is proposed to represent programming concepts as concrete objects in a virtual environment. Secondly, is the need to address misconceptions of programming concepts where a simulation design for misconceptions of programming concepts. Thirdly, is the need to incite intrinsic motivation when learning programming. To handle these issues, a framework that combines the potential of virtual reality and game elements was proposed. According to data gathered from questionnaires, we can conclude that the identified issues were mitigated accordingly.

In the future, further work can be done to complement the system. Particularly, in terms of providing more support for different programming languages. This would allow the system to be used by a wider range of audience so as not limit the learning of multiple programming languages. Furthermore, the current framework can also be used to aid understanding of more complex data structures such as trees or graphs. As for further improvements, the current framework can also be combined with other technologies such as haptics. This would aid in considerations regarding the implementation of haptics technology, particularly for education purposes. Lastly, the framework can also be adapted and revised to explain abstract topics in other domains such as engineering, science and mathematics.

References

1. Fayer, S., Lacey, A., Watson, A.: STEM Occupations: Past, Present, And Future. U.S. Bu-reau of Labor Statistics (2017)
2. Higher Education Student Statistics: UK, 2018/19 - Subjects studied, UK (2020)
3. Non-continuation: UK Performance Indicators 2018/19. Higher Education Student Statistics, UK (2020)
4. Giannakos, M.N., Aalberg, T., Divitini, M., Jaccheri, L., Mikalef, P., Pappas, I.O., Sindre, G.: Identifying dropout factors in information technology education: A case study. In: 2017 IEEE Global Engineering Education Conference (EDUCON). pp. 1187–1194. IEEE, Athens, Greece (2017). https://doi.org/10.1109/EDUCON.2017.7942999
5. Pappas, I.O., Giannakos, M.N., Jaccheri, L.: Investigating factors influencing students' intention to dropout computer science studies. In: Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '16. pp. 198–203. ACM Press, Arequipa, Peru (2016). https://doi.org/10.1145/2899415.2899455
6. Tan, P.-H., Ting, C.-Y., Ling, S.-W.: Learning difficulties in programming courses: undergraduates' perspective and perception. In: 2009 International Conference on Computer Technology and Development. pp. 42–46. IEEE, Kota Kinabalu, Malaysia (2009). https://doi.org/10.1109/ICCTD.2009.188
7. Medeiros, R.P., Ramalho, G.L., Falcao, T.P.: A systematic literature review on teaching and learning introductory programming in higher education. IEEE Trans. Educ. **62**, 77–90 (2019). https://doi.org/10.1109/TE.2018.2864133
8. Dasuki, S., Quaye, A.: Undergraduate students' failure in programming courses in institutions of higher education in developing countries a Nigerian perspective. Electron. J. Inf. Syst. Dev. Countries **76**, 1–18 (2016). https://doi.org/10.1002/j.1681-4835.2016.tb00559.x
9. Kohn, T.: The error behind the message: finding the cause of error messages in Python. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education - SIGCSE '19, pp. 524–530. ACM Press, Minneapolis (2019). https://doi.org/10.1145/3287324.3287381
10. Perron, B., Schröter, F. (eds.): Video Games and the Mind: Essays on Cognition, Affect and Emotion McFarland & Company Inc. Publishers, Jefferson (2016)
11. Pears, A., et al.: A survey of literature on the teaching of introductory programming. SIGCSE Bull. **39**, 204 (2007). https://doi.org/10.1145/1345375.1345441
12. Vincur, J., Konopka, M., Tvarozek, J., Hoang, M., Navrat, P.: Cubely: virtual reality block-based programming environment, pp. 1–2. Association for Computing Machinery (2017)
13. Bouali, N., Nygren, E., Oyelere, S.S., Suhonen, J., Cavalli-Sforza, V.: Imikode: A VR game to introduce OOP concepts. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research - Koli Calling '19, pp. 1–2. ACM Press, Koli (2019). https://doi.org/10.1145/3364510.3366149
14. Chen, J., Zargham, M.R., Rajendren, M., Cheng, J.: Coding VR games. In: Int'l Conf. Frontiers in Education: CS and CE, pp. 123–127 (2019)
15. Segura, R.J., Pino, F.J., Ogáyar, C.J., Rueda, A.J.: VR- OCKS: a virtual reality game for learning the basic concepts of programming. Comput. Appl. Eng. Educ. **28**, 31–41 (2020). https://doi.org/10.1002/cae.22172
16. Norris, E., Shelton, N., Dunsmuir, S., Duke-Williams, O., Stamatakis, E.: Virtual field trips as physically active lessons for children a pilot study. BMC Public Health **15,** 366 (2015). https://doi.org/10.1186/s12889-015-1706-5
17. Nederveen, J.P., Thomas, A.C.Q., Parise, G.: Examining the first-person perspective as appropriate prelaboratory preparation. Adv. Physiol. Educ. **43**, 317–323 (2019). https://doi.org/10.1152/advan.00213.2018