Received 12 July 2022, accepted 31 August 2022, date of publication 5 September 2022, date of current version 27 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204392

RESEARCH ARTICLE

iProgVR: Design of a Virtual Reality Environment to Improve Introductory Programming Learning

CHYANNA WEE, KIAN MENG YAP, (Senior Member, IEEE), AND WOAN NING LIM, (Senior Member, IEEE)

Research Centre for Human-Machine Collaboration (HUMAC), Department of Computing and Information Systems, Sunway University, Bandar Sunway, Petaling Jaya, Selangor 47500, Malaysia

Corresponding author: Chyanna Wee (15060197@imail.sunway.edu.my)

This work was supported in part by the Internal Research Grant Scheme (IRGS) under Grant GRTIN-IRG-135-2021.

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by Sunway University Research Ethics Committe under Application No. PGSUREC2020/030.

ABSTRACT Currently, there are a plethora of solutions developed to help students learn the basics of programming. However, there is a relative paucity of solutions that 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 motivation. Hence, in this study, a framework to address the abstract nature of programming and common programming misconceptions is developed. The framework consists of three modules that correspond to each issue, powered by a simulation engine. The first module is developed to address the abstract nature of programming by representing programming concepts with concrete objects in the virtual environment. The second module employs simulation techniques such as interactions and player perspectives to address common programming misconceptions. Lastly, the third module employs elements in the virtual environment to engage students when learning through the system. To evaluate the system, 60 participants were randomly divided into the control group (N = 30) and the experimental group (N = 30). Participants in the control group were taught using a video lecture while participants in the experimental group were taught using the developed VR intervention. Evaluation results gathered quantitatively indicated that the VR intervention was able to significantly increase programming concepts comprehension and address programming misconceptions. Participants also rated the developed VR intervention to be significantly more engaging than the video lecture.

INDEX TERMS Computing education, virtual reality, programming.

I. INTRODUCTION

The United States Department of Labour's Bureau of Labour Statistics (BLS) has predicted an increase of 12.5% in computing-related jobs from the year 2014 to 2024 [1]. This is synonymous with the growth in enrolment rates of computing courses with a 7% increase from the year 2017 to 2019 [2]. In the same report by Higher Education Student Statistics (HESA), it is found that the percentage change of computer science courses is the highest among science, technology, engineering, and math subject areas. However, dropout rates for computer science courses are also the highest among other Science, Technology, Engineering, and Mathematics (STEM) courses at 9.8% [3].

Recent studies done on reasons for the high non-continuation rates have ranged from negative experiences, low sense of belonging and poor teaching, resulting in low grades [4]. Consequently, Pappas *et al.* [5] who sampled 1050 Norwegian computer science students found that students with higher expectations often fail to fulfil them, which reduces retention. Tan, Ting, and Ling [6] and Medeiros [7] cites reasons such as the abstractness and misconceptions of programming concepts as well as student's lack of intrinsic motivation to be the root cause of failures in introductory

The associate editor coordinating the review of this manuscript and approving it for publication was Andrea F. Abate.

programming courses. Ultimately, the authors conclude that this also leads to the increase of non-continuation rates. Specifically, the abstractness of programming languages makes understanding the concepts harder since it is tough for students to comprehend how the concepts relate to real life. This in turn causes misconceptions of the programming concepts learnt, which results in syntax errors, decreasing the confidence of students, and ultimately causing them to drop out from the course.

This has led researchers to come up with ways to make introductory programming courses easier to comprehend while engaging students. This include game-based techniques, tangible tools, and visual programming tools. With the commercialization of virtual reality (VR) devices, more VR experiences have been developed to teach programming concepts in recent years. However, existing techniques that employ VR mainly focus on increasing student engagement. Thus, there is a need to go beyond just engaging students but to also address the abstractness and misconceptions that students may have regarding programming concepts. Hence, the primary research aim is to utilize the widely known benefits of VR to address the issues students face when learning introductory programming.

This paper also serves as an extension to a previously published conference paper [8]. Compared to the published article, this paper features more comprehensive coverage on previous works, extended methodology with the inclusion of selection statements, significantly expanded and more thorough experimental methodology with the inclusion of pre- and post-tests, and workload comparisons between the VR and non-VR version of the same intervention. Furthermore, this paper also features a more detailed discussion on the findings gathered from the experiments.

The paper is organized in sections, where Section II discuses work done on computer programming education, game-based learning and virtual reality tools, and workloads of virtual and non-virtual reality applications. Section III and IV highlights the methodology and experimental methodology used to gather data. Lastly, Section V showcases the results gathered, Section VI discusses the results, and Section VII concludes the paper.

II. LITERATURE REVIEW

This section highlights the main issues students face when learning computer programming. Existing work to aid in computer education are then discussed ranging from approaches that utilize VR, tangible tools, and visualization tools. The review then goes on to discuss the semantic, syntax, and analogies for computer programming. Lastly, the research gaps will be determined by comparing existing works that aim to facilitate computing education.

A. ISSUES FACED WHEN LEARNING COMPUTER PROGRAMMING

In a 2019 survey of 161 participants spanning across various continents (Africa, Asia, Australasia, Europe, North America, and South America) by Bennedsen and Caspersen [9], failure rates of introductory computer programming courses were found to be at 28%. However, the author notes that this number is subjective across different universities or colleges where elite schools will deem this number high while other schools may find this number acceptable. Becker *et al.* [10] argue that while the contributing factor for the failure rates mentioned above is multifaceted, the root problem encompassing this issue is the difficulty of learning how to program, which ultimately leads to increased dropout rates. This section aims to highlight some of the issues faced by students when starting an introductory programming course. According to Medeiros *et al.* [7], the problems faced by students are problem formulation, solution expression, solution execution and evaluation, and behaviour. However, to better situate the study, only the top issue from each category and only issues directly related to building programming knowledge will be highlighted. Hence, the main problems faced by students when learning computer programming are the abstract nature of programming, difficulties with programming syntax, and student behaviour (motivation and engagement).

These days, computer programming is mostly taught with high-level programming languages where users only need to deal with variables, arrays, and loops compared to call stacks, registers, and memory addresses. While this improves code readability and usability, users may find programming concepts to be too theoretical, making it harder for some to grasp the fundamentals of programming. This problem is also highlighted in a research by Dunican [11] who proposed analogies in introducing programming concepts to relate the concept of variables and data types to objects and scenarios in order to aid student's learning. Dasuki and Quaye [12], who interviewed 28 undergraduates from a Nigerian university found that students could not comprehend how programming concepts could solve real-world problems, resulting in disinterestedness towards programming. According to Giraffa *et al.* [13], programming concepts are hard to grasp due to the need to imagine abstract terms that do not correspond to real life objects. This issue is also prevalent in literature by Gomes and Mendes [14] and Eltegani and Butgereit [15]. Consequently, despite being the easiest data structure, arrays are often cited as being potentially difficult to understand [16], [17], [18]. This ultimately inhibits students from applying the concept when developing programs [17]. Hence, it is important to map programming concepts to real-world contexts to help students understand the fundamentals and what type of problems can be solved with programming.

Additionally, the need to simultaneously learn syntax and semantics can be overwhelming [19]. According to Giraffa *et al.* [13], students often find it difficult to convert the solution to a problem written in pseudocode into a syntactically correct program. Hence, syntax errors and helping students address them should also be considered as part of the learning process [7]. Koulouri *et al.* [19]

argued that syntax errors have significant implications on novice programmers as the process of identifying and addressing them can be time-consuming. This explains why syntax errors are widely discussed in the programming pedagogy field. Qian and Lehman [20] also covered this extensively in their review, stating that misconceptions can arise due to difficulties in both syntactic and conceptual knowledge. According to Kohn [21], misconceptions of programming concepts also cause syntax errors. Hence, syntax errors that arise due to misunderstanding programming concepts like variables, loops, arrays and if statements will be highlighted.

To start, variables correspond to labels which are given to specific memory locations that store data. Being the most essential concept in computer programming, failure to understand the concept of variables will lead to problems as every other concept requires the application of variables. However, the most common misconception relating to variables is that students assume variables can store more than one value. This is seen in early studies carried out by Sleeman *et al.* [22], Doukakis *et al.* [23] and a recent study by Swidan *et al.* [24]. Secondly, students assume that the order of statements when assigning a value to the variable is not important. This is also argued by early studies done by Ma [25], Sirkia and Sorva [26] and a recent study by Kohn [27]. Older studies are included in this section to highlight that these misconceptions exist throughout time, regardless of programming languages. Arrays are extensions to the concept of variables except for being able to store multiple values at once. This is done by assigning index numbers to individual data values present in the array. While this is necessary for effective data retrieval, the fact that indexes start with zero is particularly confusing for those just starting to program [28]. Whittall *et al.* [29] argue that programs like Scratch exacerbates this confusion by indexing the first element of the array as one instead of zero.

Consequently, programming is perceived by many as an intimidating and difficult task [29]. This can be due to the way computer programmers are portrayed in the media, in which they are categorised as "nerdy" and "geeky" which corresponds to being intelligent [30]. For those who have never programmed before, this may seem like a reason to be intimidated. For those that eventually enrol in introductory programming courses, feelings of anxiety may arise when one finds that he/she is not "smart" enough to program. This is further proved by a study carried out by Chang and Sheeson [31] who surveyed 307 participants, showed that the relationship between how an individual perceives the difficulty programming and anxiety levels is directly proportional. This is crucial as one's perception of programming and one's perceived ability to be a good programmer significantly affects their programming skills [32]. For this reason, motivation in acquiring programming skills and knowledge may also dwindle, further hampering student engagement.

B. VIRTUAL REALITY FOR COMPUTER PROGRAMMING EDUCATION

The use of virtual environments for teaching is supported by the constructivist theory [30], [31], [32] and the embodied congnition theory [32],[33]. Hence, this section highlights existing techniques that employ virtual reality techniques in developing experiences for use in learning computer programming. Tanielu *et al.* [34] developed a VR experience called "OOPVR" to reduce the abstractness of OOP concepts with analogies. To increase a student's understanding of objects and classes, OOPVR utilizes a house to depict a class where multiple houses can be built using the corresponding blueprint. Thus, this implies how a class can instantiate multiple objects. Throughout OOPVR, many analogies were implemented to represent programming concepts namely instances, methods, and encapsulation. To determine the effectiveness of the system, the authors analysed results from the pre- and post-surveys that probed 17 students on how confident they were when it comes to visualizing various OOP concepts. Results revealed that participants were significantly more confident in conceptualizing OOP concepts after the experience.

Similarly, Singh [35] proposed a VR experience in which students can learn basic programming concepts by visualizing code. Students are presented with a code block in which they will be able to highlight specific lines of code or parts of a line of code to observe how the code is represented in the virtual space. For instance, if the student highlights the code "int x = 10;" they will be presented with a cube labelled with the name of the variable (in this case, the variable name corresponds to "x") and a sphere to represent the value (in this case, the value corresponds to 10). Students are also required to place the sphere into the cube to represent the assignment of a value to a variable. However, there were no testing done to prove the effectiveness of the VR experience.

Vincur *et al.* [36] combined both VR and game elements with block-based programming to develop "Cubely", which is used to teach programming. Cubely contains cubes that represent programming that can be arranged to construct programs. Students are required to build programs from code blocks to control and guide their character to complete challenges. According to the study's results, out of 19 participants, 18 preferred Cubely because it is easier to use and is more engaging than typical online code bootcamps like Code.org. Bouali *et al.* [37] developed a VR game called "Imikode" to help students familiarise themselves with OOP concepts. This system provides an opportunity for students to create and build virtual worlds using programming. For instance, students can use commands like "fox = new Fox()" to create a fox into their virtual world. Despite that, no testing was conducted to confirm the system's effectiveness on students.

Consequently, Chen *et al.* [38] developed a VR tool that allowed students to create levels in a game to challenge their peers. Before starting the game, the student playing the role of the level creator will write codes in the virtual environment to place robot characters around the environment that will act as obstacles. Students who then play the game are required to acquire hints scattered around the virtual environment while overcoming the obstacles set up by the previous student. In a sense, students will be able to judge the effectiveness of their own code depending on how the system reacts. The authors claim that students of age 9 to 13 who tried the game have provided positive feedback in terms of engagement and learning effectiveness. Similarly, Segura *et al.* [39] developed a VR game called ''VR-OCKS'' that required students to use code functions represented with blocks in the game to complete puzzles. To test the effectiveness of the system, 20 participants that have played VR-OCKS and another 20 participants who didn't play the game were recruited. Each group of 20 participants was further halved to complete challenges synonymous with the puzzles presented in VR-OCKS in ''Kodu'' and ''Blockly'', two popular systems that utilise visual programming to help students learn programming. Results indicated that those who played VR-OCKS before were able to complete 25% more levels than those who didn't.

Stigall and Sharma [40] developed a VR game to teach OOP concepts such as polymorphism, inheritance, and encapsulation. For instance, to teach the concept of inheritance, students are required to build a vehicle. The vehicle can be built with a blueprint, where child classes containing specific features will extend the Vehicle class. The module was tested with the aid of 15 undergraduate students, and it was found that 92% of students found the system helped them learn OOP concepts. Consequently, Kao *et al.* [41] developed a game called ''HackVR'' to also teach students OOP concepts. HackVR features nodes, which contain programming constructions such as objects, conditionals, function calls, and event handlers. Students are required to build programs with a set of the mentioned nodes. For instance, an entity node represents objects in the virtual world (door), programs attached to the node will then trigger the virtual object (opening the door). The authors expect that the system might aid in the teaching of programming concepts. However, no tests have been done so far to confirm this.

Jin *et al.* [42] developed ''VWorld'', a tool that allows children to create virtual worlds and program 3D objects. The objects in the virtual environment can be programmed by utilizing blocks that represent instructions to form a code sequence. Testing was carried out with 3 university students, and it was observed that the tool was both engaging and easy to use. Zargham and Kamsani [43] developed a VR tool that teaches programming by making students design and program a roller coaster in the virtual environment. Students can also observe their design in VR to fix bugs or add more features to their roller coasters. To test the feasibility if the system, 7 participants from ages 10 to 15 were recruited. It was observed that the tool was engaging to use. Similarly, Parmar *et al.* [44] designed and developed ''Programming Moves'', to teach programming by allowing students to program an avatar to carry out dance moves. The avatar is programmed by dragging and dropping pre-programmed moves to form a programmatic sequence. 47 participants were recruited to test the system and it was found that students though the tool was immersive, easy to use, and engaging.

C. TANGIBLE AND VISUAL TOOLS FOR COMPUTER PROGRAMMING EDUCATION

Aside from VR applications for computer programming education, researchers have also come up with solutions which feature visual representations that are also tangible. For instance, Sabuncuoğlu *et al.* [45] developed a tangible tool that makes use of cardboard blocks that correspond to commands. With this, students arrange the blocks to create algorithms. This is also complemented with the development of an Android app that scans the blocks and compiles them into actual programs that can trigger various events. The tool was tested with students that were between 12 to 13 years old who attended an introductory programming course. Results showed that the tool managed to improve spatial reasoning and programming skills. Consequently, González [46] utilised tangible blocks to identify novice's misconceptions in programming. This is done with the development of an Android application that scans the blocks that are arranged by the students to form algorithms and checks it for any errors. The feasibility of the system was tested with 9 students who have taken an introductory programming course. While the system was well received by the students, the authors claim that the feedback system needs to be improved further as students tend to ignore some of the information displayed and thus missed out on learning.

Furthermore, some researchers also employed visualisation techniques to help students learn and program better. For instance, Lerner [47] developed an extension for the Visual Studio Code IDE that displays projection boxes which features code visualizations. These visualisations mainly show the state of various data structures in a program during code runtime. The tool was tested with the aid of 10 participants whose experience ranged from medium to expert. Results showed that the tool helped users write code correctly with an average score of 4.7 on the Likert scale. Similarly, Mladenović *et al.* [48] used a tool called ''Python Tutor'' to help students understand what happens as the computer runs each line of code in a program. The authors carried out experiments with the aid of 98 students where 44 students were placed in the control group while 54 students were placed in the experimental group that was taught with various visualisation tools. Results showed there were no significant differences between both groups. The authors claimed that while the results suggest that visualizations can help students understand programming concepts, it should only be used as a supplement in lectures to promote student participation during lessons. Consequently, Khaloo *et al.* [49] developed a 3D code visualisation tool that aims to improve students understanding of programming concepts. To do so, the visualisation tool represents each class as a separate 3D environment and lays out data structures and functions on the walls. The system was tested with the aid of 28 university students and results showed that the tool was able to improve code understanding.

D. VISUAL PROGRAMMING TOOLS

Visual programming allows users to freely create and manipulate programs with a graphical interface rather than just lines of code [50]. This makes programming a less daunting task as it focuses on how to solve a particular problem instead of the syntax of a particular programming language [51]. This, in turn, allows students to foster their problem-solving skills even with the restriction of a time limit. Compared to Textual Programming Languages (TPL), Visual Programming Languages (VPL) can also foster positive attitudes towards learning programming [52].

Block-based programming applies the principles of visual programming as it involves blocks representing certain ''code blocks'' that exist in a typical computer program. This means that users can program with these blocks, providing them a visual and graphical interface when doing so. Some examples of block-based programming applications include Scratch, Alice, and Blockly. Primarily developed for children, Scratch allows users to create online projects with a drag and drop block-based interface [53]. Similarly, Alice allows users to develop and build computer animation with three-dimensional models [54]. Blockly, developed by Google is also similar, except it is used in applications where developers are provided the syntax and programming representation to make application building a more seamless process [55]. However, not all visual programming applications are drag and drop block based. For instance, Pure Data [56] incorporates flow and state diagrams while Kodu uses icons [57].

Aside from making programming less daunting, visual programming languages can also increase learners' motivation [58]. According to Seraj *et al.* [59], when surveyed if blocks or code were preferred for programming, students were found to favor blocks with more girls opting for blocks than boys. Consequently, Milo (similar to Scratch but used for machine learning classes) by Rao *et al.* [60], found that 90% of the participants agree that block-based programming made understanding concepts easier. However, the drag and drop block-based interface of these programs can pose a program as it generally takes more time to drag blocks than typing [61].

E. GAME-BASED TOOLS FOR PROGRAMMING

Game-based tools in this context refer to applications that are developed to aid in computing education, specifically computer programming. In a review done based on the achieved outcomes of programming games by Lindberg *et al.* [62], it is found that when game elements are implemented in the context of computer programming education, outcomes like learning effectiveness, motivation, and engagement can be achieved.

Papadakis and Kalogiannakis [63] utilized Classcraft, an online platform to manage student behaviour and learning. The platform employs gamification techniques such as levelling up, earning powers, working in teams, and allows teachers to set up programming-based quests and challenges for their students. By conducting surveys on 30 participants from a high school in Greece, it was found that participants in the experimental group were engaged and motivated throughout the experience. However, it was also found that there was no significant difference between the control group and the experimental group in terms of students' performance for a test. Furthermore, Mathrani *et al.* [64] utilized a game called ''LightBot'', to test the feasibility of the system in regards to teaching programming concepts. In the game, participants are required to guide a robot to solve puzzles using blocks that represent commands identical to code. Evaluation results gathered from 20 participants found that the game successfully introduced programming concepts such as recursion, conditionals, and functions.

Wong and Yatim [65] developed a game called ''The Odyssey of Phoenix'' to aid in the learning of Object-Oriented Programming (OOP) concepts. This is accomplished by mapping game processes to their respective concepts. The inheritance concept, for instance, is mapped into the game's crafting element, where resources needed by both the nose and main gear can be shared due to them falling under the gear category. Results gathered from the pre-and post-test of 214 first-year university students were analysed to determine the game's effectiveness. It was revealed that there were substantial differences between pre-and post-test results, implying that GBL is an excellent tool for knowledge acquisition and learning. Moreover, Oyelere *et al.* [66] created the ''MobileEdu-puzzle'' game to also teach programming. The game functions by requiring students to arrange disorganized lines of code. It was found that 71% of the 51 students who took part in the study said that they were able to learn programming effectively because of the experience. However, no pre- and post-test were carried out to support this fact. The participants also responded positively when asked if the game managed to motivate them to learn programming.

Orehovački and Babic [67] studied the feasibility of ''CodeCombat'' in which participants had to develop and execute Python code to solve quests. Typically, the code is written to move a character from one point to another. Results gathered from 175 participants showed that students had a positive attitude towards the game to learn programming concepts. Durán *et al.* [68] developed a game called PLMan where students are required to build controllers with the Prolog programming language. The goal is to develop a controller with a set of basic rules such as ''If a ghost is at your right, move left'' to enable the game character to devour all the dots placed in the maze-like environment. The game was employed as part of a computational logic class for two years during the year 2015 to 2016 with an average of 300 students each term. Results showed that almost 70% of the students had chosen computational logic as their favourite subject.

Daungcharone and Panjaburee [69] developed a game called ''CP m-Game'' to teach students programming in the C language. The game works by presenting students with a narrative in which they were asked to analyse the problem and develop a solution for the problem by developing code out of code blocks. The game would also guide students if there were any errors in their code. To evaluate the effectiveness of the system, a total of 50 participants were recruited. Participants were asked to complete a pre-test in the form of a lab assessment before trying out the intervention and a post-test after. Results showed that students performed significantly better for the post-test, were more motivated, felt more confident in terms of the C programming syntax, and were confident when doing the tests. Min *et al.* [70] developed a game called ''Engage'' to foster computational thinking via a narrative-based game. In the game, students are required to restore a faulty server network in an undersea research facility. This can be done by developing code with a block-based programming language to solve puzzles. The feasibility of the system was tested with the aid of 14 participants who completed a post-test survey, and it was found that the game was highly engaging.

Kazimoglu [71] developed a game called ''Program Your Robot'' to increase student motivation and confidence when it comes to programming. The game is like ''Lightbot'', in which students are required to get a character from one point to another by using a series of commands. The feasibility of the system was tested with the aid of 151 students. It was found that students showed significant improvements in terms of intrinsic motivation, programming knowledge, and confidence after the intervention. Malliarakis *et al.* [72] developed a game called ''CMX'' to determine the effects of a Multiplayer Online Role-Playing Game (MMORPG) to teach and learn computer programming. Throughout the game, students are introduced to programming concepts by an in-game character. The knowledge gained can then be used to accomplish various challenges set up in the game. The system was tested with the aid of 76 participants who were placed in the experimental group while 234 participants were placed in the control group. Results showed that students who utilized the game had a higher mean score during their midterm exams. It was also found that students found the game to be engaging.

Bonner and Dorneich [73] developed a game called ''Sorceress of Seasons'' to increase female participation in the field of computer science. Students are required to complete challenges by using Python code to command an in-game character. The system was tested with 15 middle school students. Results showed that female participants were more likely to choose a computer science-related field after the experience. According to Sharma *et al.* [74] in their review, it was found that serious games for computer education can improve girls' perception of computer science as a viable career option.

F. SEMANTIC AND SYNTAX OF COMPUTER PROGRAMMING

According to Shneiderman and Mayer [75], a complex multi-layered body of knowledge is developed in every experienced programmer about programming concepts and techniques. Part of that body of knowledge includes both semantic and syntactic comprehension. Semantic knowledge ranges from low-level notions of what subscripted arrays and data types are or how assignment statements work; to more intermediate notions such as summing up all of the elements that are present in an array or developing an algorithm to find the larger of two values; to high-level notions such as recursion and sorting. On the other hand, syntactic knowledge ranges from familiarising oneself with the valid character sets of a conditional or assignment statement, the format of an iteration, and the names of library functions. To ensure students can master both semantic and syntactic knowledge in tandem, Shneiderman [76] proposed the spiral approach. This approach follows the cognitive model, which works by presenting students with small amounts of semantic and syntactic knowledge at a time. Each time, the new knowledge presented should contain both semantic and syntactic components, should contain minimal addition and be related to the previous knowledge, and presented with meaningful and relevant examples. Existing tools for computer programming education can be grouped into two categories in terms of the approaches employed, the first being the construction-first approach and the second being the comprehension-first approach. The construction-first approach is an iterative process that works by making programmers write code, build, and run the code, observe the output, and then revise the code [77]. On the other hand, the comprehension-first approach prioritises building a mental model of programming semantics to build intuition which can then be used to build programs with fewer semantic-based misconceptions [78].

G. ANALOGIES FOR THE VARIABLE DATA STRUCTURE

The mapping between similar characteristics of unrelated principles or concepts is known as analogies [79]. In theory, an analogy used during a lesson should be familiar and should have a large degree of correlation between the source and the concept that is targeted. This is because familiar analogies promote learning [80] and increases student engagement. However, it is also important to note that since both the analogy and the concept are not identical, a perfect correlation is not possible. The process of teaching and learning computer programming is a tedious task as it requires significant cognitive, abstraction, logical, and mathematical skills [81]. Due to this, multiple studies have been dedicated to investigating the effects of utilizing analogies in the teaching and learning of computer programming [65], [34], [35], [82].

According to Chibaya [83], Since programming revolves around manipulating and managing data, lessons should incorporate mind-maps or visualizations of memory processes from when data is inputted, when data is processing, and when data is outputted. Therefore, it is important to have students visualize the computer memory as a large storage space capable of storing many pieces of data. This means that the addition, management, tracking, and traversing of data elements should be closely thought of as the act of programming. Variables are typically defined as labelled spaces in memory where data is stored [84], [85] or containers of data elements stored in memory [11], [86], [87]. Furthermore, the process of naming and defining a variable is called a declaration, and often, newly declared variables are thought of as empty containers to be filled with a data value. When the need arises, the data value can also be modified. This explains why the process of inputting and outputting data is often likened to the act of filling a container and the act of retrieving the content from the container respectively [88]. Hermans *et al.* [89] used the box metaphor for teaching students the concept of variables. Results show that the box metaphor is suitable to help students understand the basic concept of variables.

H. SUMMARY

As seen in Table 1, most VR applications for computer programming education mainly focuses on promoting student engagement. As mentioned in section A, the main issues students face when learning programming range from the abstract nature of programming, programming misconceptions, and lack of motivation. However, there is a relative paucity of studies that directly tackle the issues of the abstract nature of procedural programming and programming misconceptions. Furthermore, in terms of methodology, there is also a paucity of tools that visualize memory systems which incorporate representations of programming concepts as concrete objects in VR. Consequently, there is also a paucity of tools that addresses programming misconceptions via interacting with the environment in VR. This is significant as work done in this study will provide a one stop solution to mitigate the main issues faced by students when learning programming as opposed to existing studies that only tackle one issue at a time. Thus, this study aims to expand the already well-known benefits of VR, to do more than engage students when learning programming.

One common observation is the lack of comprehensive testing, as most works in this field are limited to conference papers [90]. Hence, the experimental methodology for this study will range from pre- and post-assessments, and surveys to access perceived outcome. Moreover, since there is also a paucity of studies that compared the workload of VR and non-VR interventions in the context of computer programming, this study will also address this gap.

With that said, the following hypothesis corresponding to each issues students face when learning programming are proposed:

1. To mitigate the abstract nature of programming concepts, the following ana is proposed. "Abstract programming concepts are easier to understand if it is represented as concrete objects in the virtual environment".
2. To address programming misconceptions, the following hypothesis is proposed. "By allowing students to interact with the virtual environment, programming misconceptions can be better identified and addressed".
3. To increase student engagement and motivation, the following hypothesis is proposed. "The developed intervention will improve students' motivation and engagement when learning computer programming".

TABLE 1. Comparisons of existing VR tools for teaching computer programming.

Ref	Methodology	Testing	Outcome(s)
[34]	Visualized OOP concepts.	Survey	Reduce abstractness for OOP.
[35]	Visualized programming concepts.	No tests done	No reported outcomes.
[36]	Build programs out of code blocks to solve challenges.	Survey	Motivation and engagement.
[37]	Build virtual worlds with code.	No tests done	No reported outcomes.
[38]	Build game levels with code.	Survey	Motivation and engagement.
[39]	Solve puzzles with code blocks.	Survey	Motivation and engagement.
[40]	Mapped OOP concepts to game processes.	Survey	Reduce abstractness for OOP.
[41]	Solve challenges with nodes that represent programming commands.	No tests done	No reported outcomes.
[42]	Program virtual objects.	Not mentioned	Motivation and engagement.
[43]	Program roller coasters in VR.	Not mentioned	Motivation and engagement.
[44]	Program avatars to carry out dance moves.	Survey	Motivation and engagement.
Propo sed	Visualize memory systems by incorporating representations of programming concepts as concrete objects in VR.	Pre- and post-assessment s, survey, NASA task load index.	Expected to mitigate the abstract nature of programmin g concepts, misconceptio ns, and increase engagement.

Along with the hypothesis, research questions listed below are also posed for this study:

1. How will the representation of programming concepts as concrete objects in the virtual environment aid in students' understanding of abstract concepts?
2. How will the ability to identify and address common misconceptions of programming concepts by interacting with the virtual environment aid in reducing misconceptions?
3. Will the developed intervention increase student engagement when learning computer programming?

Since there is also a relative paucity of studies that compared the VR and non-VR version of the same intervention in the context of teaching and learning computer programming, this study would also aim to carry out such comparisons.

III. METHODOLOGY

Figure 1 is a general model/framework of the proposed methodology which is divided into modules that aim to tackle programming abstractness and misconceptions. The proposed methodology employs VR elements such as the ability to represent programming concepts with concrete objects that are interactable.

Interaction in the proposed system is made possible by representing the left and right hand as aliases to the controller. With these aliases, natural hand motions like grabbing and placing objects and visual interaction cues like snap zones are possible. This, in turn, enables the development of a first-person perspective simulation, that allows students to identify common programming misconceptions, forming the misconception module. More specifically, by allowing students to interact with objects, this lets students identify and address programming misconceptions in the VR environment. This is explained further in upcoming sections below.

The abstractness module is made possible by representing variables as drawers, lists as a row of drawers and 2D arrays as lockers in the virtual environment. This module aims to reduce the abstractness of programming concepts. For now, the abstractness module mainly deals with data structures due to time constraints and to adhere to the scope of this study. Lastly, motivational elements such as a narrative context, providing instantaneous feedback and the presence of an achievement system make up the motivation module for inciting motivation in students throughout the duration of the experience. It is also important to note that the system developed based on this module is not meant to replace actual lectures but to only complement teaching.

A. MITIGATING THE ABSTRACT NATURE OF PROGRAMMING CONCEPTS

According to Barry and Griffiths [88], the process of inputting data is often likened to the act of filling a container while the process of outputting data is often likened to the act of retrieving the content from the container. Hence, it was conjectured that drawers may fit this description and can be used to represent variables in the virtual environment. Data structures that consist of a collection of elements that are identified by an index are known as arrays [91]. An array with a single dimension is known as list and they are often visualized vertically [91]. Due to the lack of literature surrounding the best way to represent lists, a row of drawers, are conjectured to fit this description and can be used to represent lists in the virtual environment. Two-dimensional array is commonly known as a table [91]. Due to a lack of literature surrounding the best way to represent 2D arrays, the locker is conjectured to fit this description as lockers have rows and columns. As mentioned, Chibaya [83] proposed that programming lessons should incorporate visualizations of the computer memory as a large storage space capable of storing many pieces of data. Hence, it is conjectured that the choice of objects like drawers and lockers can help students visualize computer memory as a large storage space. Furthermore, as seen in Table 1, existing works that surround VR applications for computer programming has yet to present programming concepts in the context of a large storage space. Hence, the development of this module effectively addresses this gap.

Figure 2 shows the system screenshot of how programming concepts are represented in the virtual environment. Drawers represent variables, a row of drawers represents lists while lockers represent 2D arrays. The name of the data structure can also be seen on the monitors above the drawers and lockers to better aid understanding.

Figure 3 shows how a challenge is presented and completed. According to the spiral approach proposed by Shneiderman [76], small amounts of semantic and syntactic knowledge should be presented at a time to ensure students can master both semantic and syntactic knowledge in tandem.

Hence, the assignment statement presented as the challenge (1) will cover the syntactic part while the act of placing the data blocks (2) into the corresponding drawer (3) will cover the semantic part of the spiral approach. 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". This simulates how an assignment statement works in the VR environment.

Similarly, lists and 2D arrays are also presented as assignment statements, requiring students to assign values to the data structures in VR. This is done by placing data blocks in a row of drawers for lists and lockers for 2D-arrays. Furthermore, the system will use Python code when presenting challenges to students since Python is perceived as easier and more

B. MITIGATING MISCONCEPTIONS OF PROGRAMMING CONCEPTS

As mentioned, tangible tools developed for programming education can help improve programming skills. This fact is also supported by the constructivist learning theory which

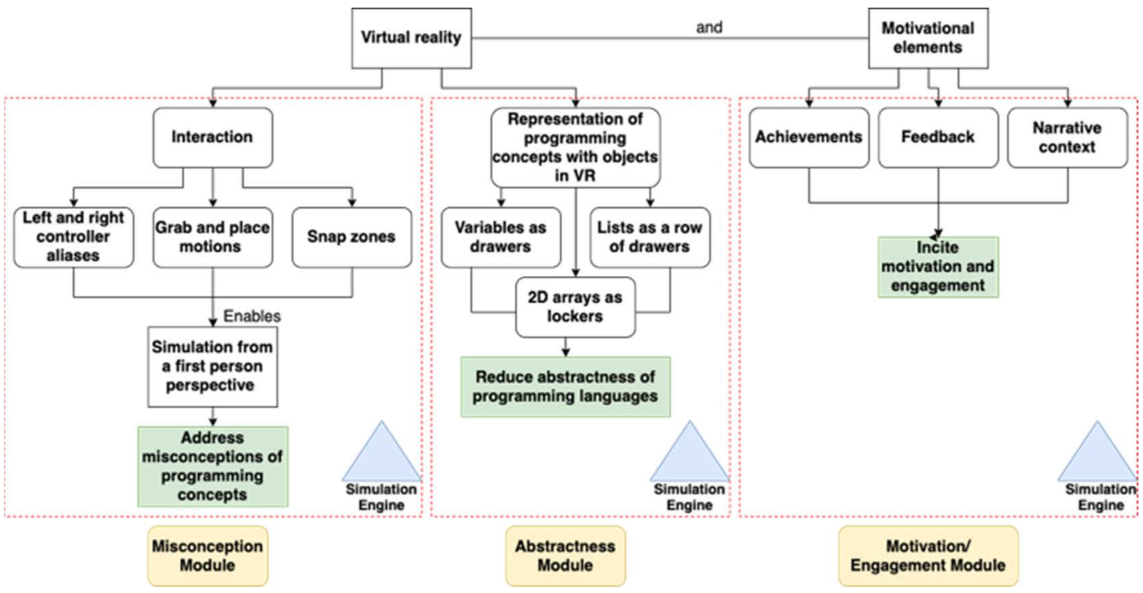

Virtual reality
and
Motivational elements
Interaction
Left and right controller aliases
Grab and place motions
Snap zones
Enables
Simulation from a first person perspective
Address misconceptions of programming concepts
Simulation Engine
Misconception Module
Representation of programming concepts with objects in VR
Variables as drawers
Lists as a row of drawers
2D arrays as lockers
Reduce abstractness of programming languages
Simulation Engine
Abstractness Module
Achievements
Feedback
Narrative context
Incite motivation and engagement
Simulation Engine
Motivation/ Engagement Module

FIGURE 1. Framework of proposed methodology.

speculates that the construction of knowledge takes place through constant interaction with the learning environment [30], [31]. Hence, by allowing students to interact with the virtual environment from a first-person perspective, programming misconceptions can be addressed. This is also supported by Winn [92] who states that immersive VR enables students to gain first-person experiences which is crucial for knowledge attainment. Winn also states that these experiences cannot be attained from formal education as schools tend to promote third-person experiences despite having first-person experiences make up the bulk of an individual's daily interaction with the world. Figure 4 shows an example of how misconceptions are handled in the virtual environment. For example, if the code presented to the user is total = 10,20, students must grab the error message block labelled with ''variables only hold one value'' and place it onto the output station. This is to highlight that variables can only store one instead of 2 values, which is a common misconception that students have [22], [23], [24]. Another common misconception highlighted in the developed VR experience are the starting index for both list and 2D arrays, which is commonly regarded as 1 instead of 0 [28], [29]. Hence, this module also addresses the gaps from existing works, where misconceptions have never been addressed in VR (Table 1). 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 ''10'' and ''20'' into the drawer labelled ''total'', the system simply does not allow this to happen. To enforce that indexes in computer programming starts from 0, students will be asked to retrieve the first element from a given list and 2D-array. Similarly, the system will not allow students to progress if the correct data block is not placed on the output station.

C. INCITING STUDENT MOTIVATION

As mentioned, the presence of a narrative context, a reward system, and visual and audio feedback in games can incite engagement and motivation in players [93], [94]. Hence, the system features all the aforementioned elements. Figure 5 shows some of the trophies that are part of the reward system in the virtual environment. Furthermore, narrative context is also presented to users during the start of the experience to foster motivation and engagement.

Users will first be presented with a narrative context before the first task. Depending on how the user completes the task presented to them, instantaneous feedback will be given in the form of audio cues. If the user completes several challenges in a row correctly, rewards would also be presented. Hence, it can be concluded that the system employs narrative contexts, a reward system and instantaneous feedback to incite learning motivation.

D. CONTROL STRUCTURES

Control structures were also included in iProgVR. Control structures like if statements and loops are presented in a similar fashion, where syntax is introduced first, followed by semantics as proposed by Shneiderman [76]. For example, if the code presented to the user is an if statement as seen in Figure 6 (1), students are asked to determine if the value in the variable ''age'' is larger than ''20''. If the value stored in the variable is above 20, it is expected that students will have to grab the ''message block'' labelled with ''you are too old'' in the virtual environment and place it on the output station. By enforcing the action of executing the if-statement in a first-person simulation, students can understand the semantics of an if-statement.

While there are many types of repetition statements, for-loops and nested for-loops were chosen for this study as

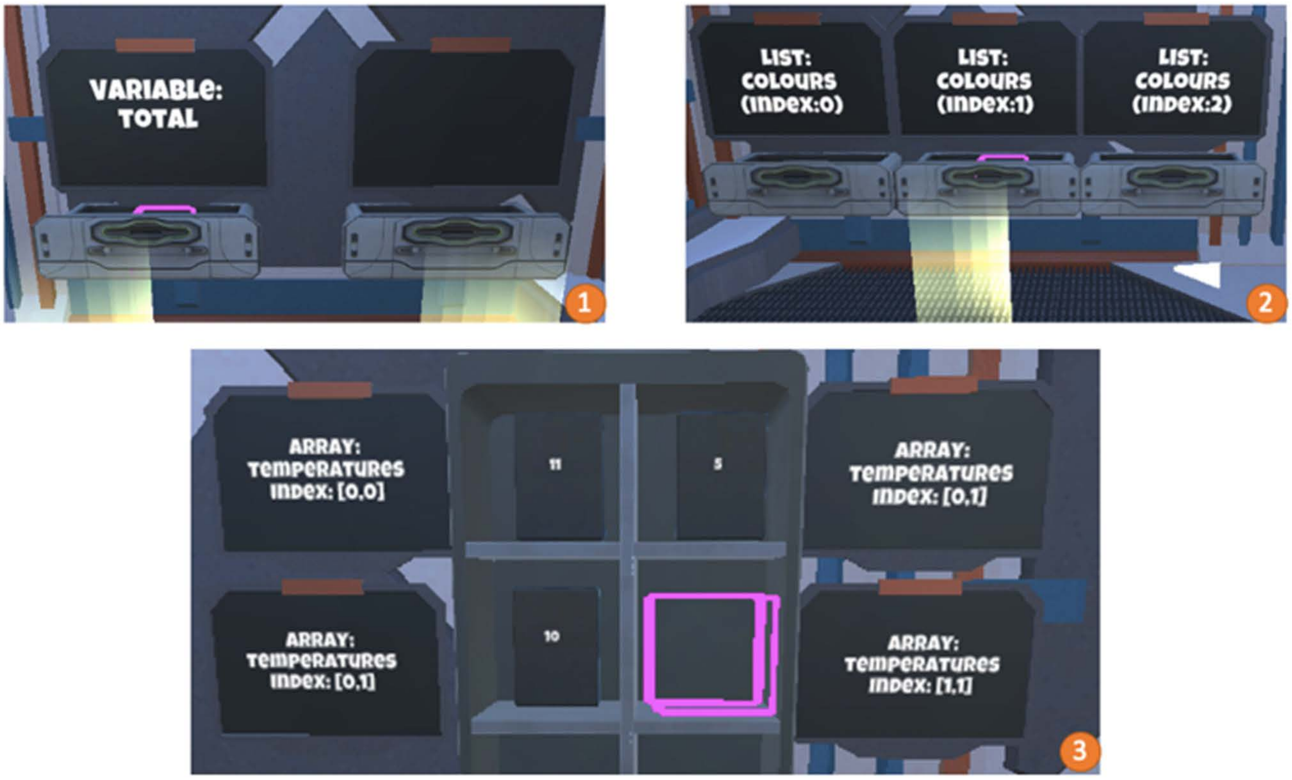
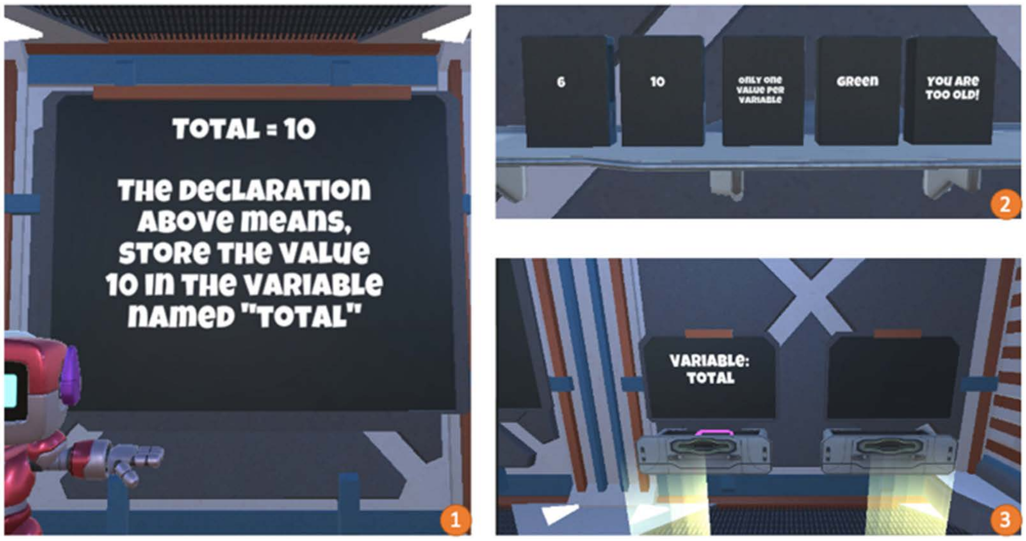

VARIABLE: TOTAL
1
LIST: COLOURS (INDEX:0)
LIST: COLOURS (INDEX:1)
LIST: COLOURS (INDEX:2)
2
ARRAY: TEMPERATURES INDEX: [0,0]
11
5
ARRAY: TEMPERATURES INDEX: [0,1]
ARRAY: TEMPERATURES INDEX: [0,1]
10
ARRAY: TEMPERATURES INDEX: [1,1]
3

FIGURE 2. (1) Drawers represent variables, (2) Row of drawers represent lists, (3) Locker represents 2D arrays.


TOTAL = 10

THE DECLARATION ABOVE MEANS, STORE THE VALUE 10 IN THE VARIABLE NAMED "TOTAL"

FIGURE 3. Steps taken to complete a challenge (from the top): (1) Assignment statement presented as the "challenge", (2) Data blocks represent values, (3) Drawer representing the variable.

it was found that students found it particularly difficult to understand the semantics of traversing 2-dimensional arrays [16]. Hence, before presenting how 2D arrays are traversed with nested for-loops, students were first presented with how lists are traversed with for-loops. Figure 7 depicts how loops are presented in the virtual environment. Similarly, the syntax for a for-loop is presented to the user, as seen if Figure 7(1). Essentially, every line of statements present in the for-loop is explained. For instance, the line "for row in colours" is explained in such a way that for every element present in the list named colours, store the value in variable "row". Then, students will be asked to observe the values stored in variable "row" at every iteration. At the first iteration, the value "blue" will be seen in variable "row", depicted by the drawer labelled "row". At the second iteration, the value "green" will be seen in variable

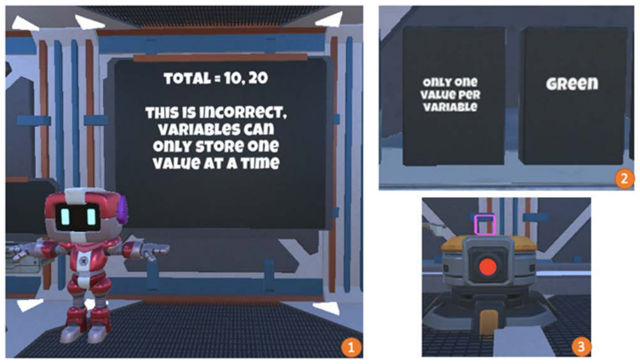

TOTAL = 10, 20
THIS IS INCORRECT, VARIABLES CAN ONLY STORE ONE VALUE AT A TIME
ONLY ONE VALUE PER VARIABLE
GREEN
1
2
3

FIGURE 4. Misconception example (from the top): (1) Code presented to user is syntactically incorrect, (2) Error message blocks, (3) Output station where error message blocks are placed.

"row". At every iteration, students will be asked to grab the value blocks labelled with "blue" and "green" on the output station to simulate how the print statement works in the context of a for-loop. Similarly, the nested for-loops were also presented by visualizing how each line works, along with a first-person simulation of how it affects the computer's memory.

E. VR DEVELOPMENT

The Unity Real-Time Development Platform, particularly version 2019.3 is chosen for the development of this experience. Developed by Unity Technologies, the engine is primarily used to develop solutions for fields like gaming, manufacturing, animating, architecture, and education. This platform was also chosen since it has been established long enough to feature a large online community. Hence, there are many resources like tutorials and discussion forums to help one get started on developing virtual reality experiences. 3D models of objects in the VR experience were imported to Unity 3D as game objects. These 3D models are mostly gathered from the Unity Asset Store. With excellent integration to Visual Studio, the system supports development with the C++ programming language. Lastly, the ability to port developed experiences to various platforms and the support for multiple virtual reality devices also influenced the choice for working with this engine.

Acquired by Facebook, Oculus has developed various virtual reality devices like the Rift, Rift S, Quest and Go. Particularly, the Rift is chosen as the device for the development of this experience due to accessibility. To develop Oculus experiences in Unity, the Oculus Integration Framework and Virtual Reality Toolkit (VRTK) are used. The Oculus Integration allows for the support of oculus devices, along with scripts that manage audio, avatars, spatializers, and prefabs for efficient development. Along with this, VRTK provides common virtual reality elements and scripts that support the implementation of interactions, user interfaces and locomotion. With the combination of both frameworks, the development of virtual experiences can be done easily and more efficiently.

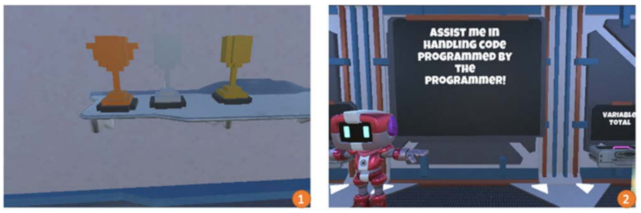
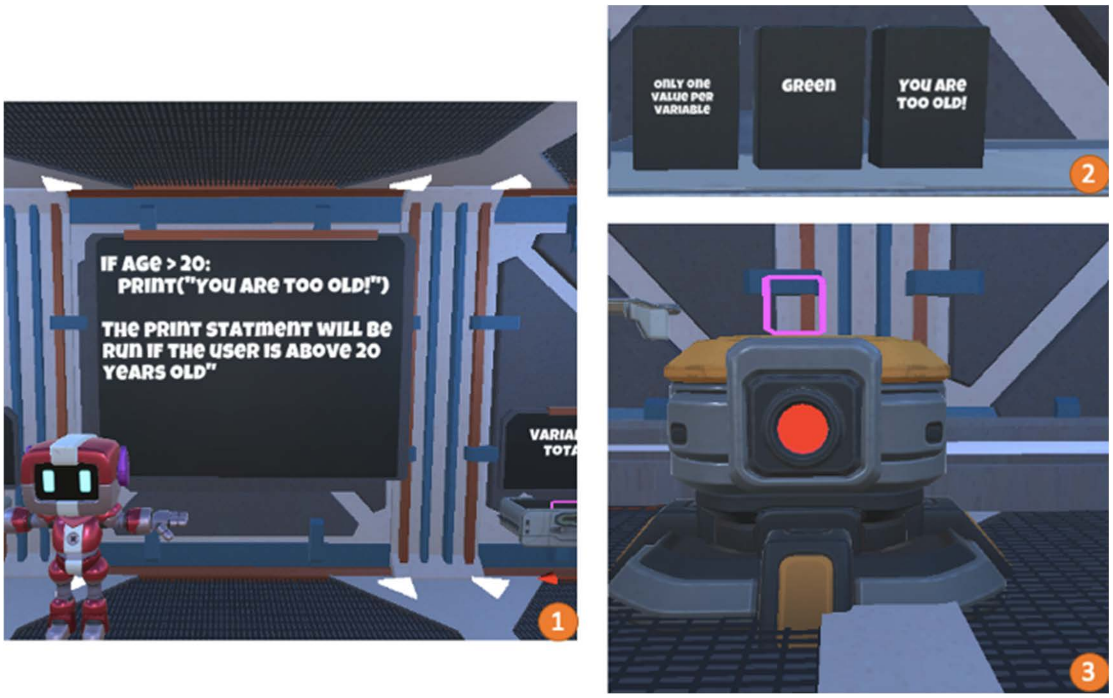


FIGURE 5. Motivational elements: (1) Trophies that can be earned during the simulation, (2) Narrative context presented to users.

In the VR environment, game object behaviours are controlled by scripts. The scripts are added to 3D models such as the "data blocks" which defines the behaviour of the object whenever it is interacted with. All scripts are written in C# language and are subsequently imported into the Unity Editor and attached to the 3D objects. VRTK snapzones are used in conjunction with scripts written to dictate the sequence of events. The scripts check to see if the user has correctly carried out the correct actions (i.e. Placing the "data block" in the correct spot) and events such as the playing of positive audio feedbacks, appearance of trophies and confetti, deactivation of previous challenges, and activation of next subsequent challenges will be set in motion. To create a simulation that is based on a first-person perspective, the tracking origin type of the VR camera rig is set to the floor level to fix the user's position and orientation relative to the floor. A model of human hands was also added with scripts to enable interaction with the Oculus Touch controllers. Additionally, control scripts also dictate the function of each button on the controllers.

F. ALTERNATIVE METHODOLOGIES

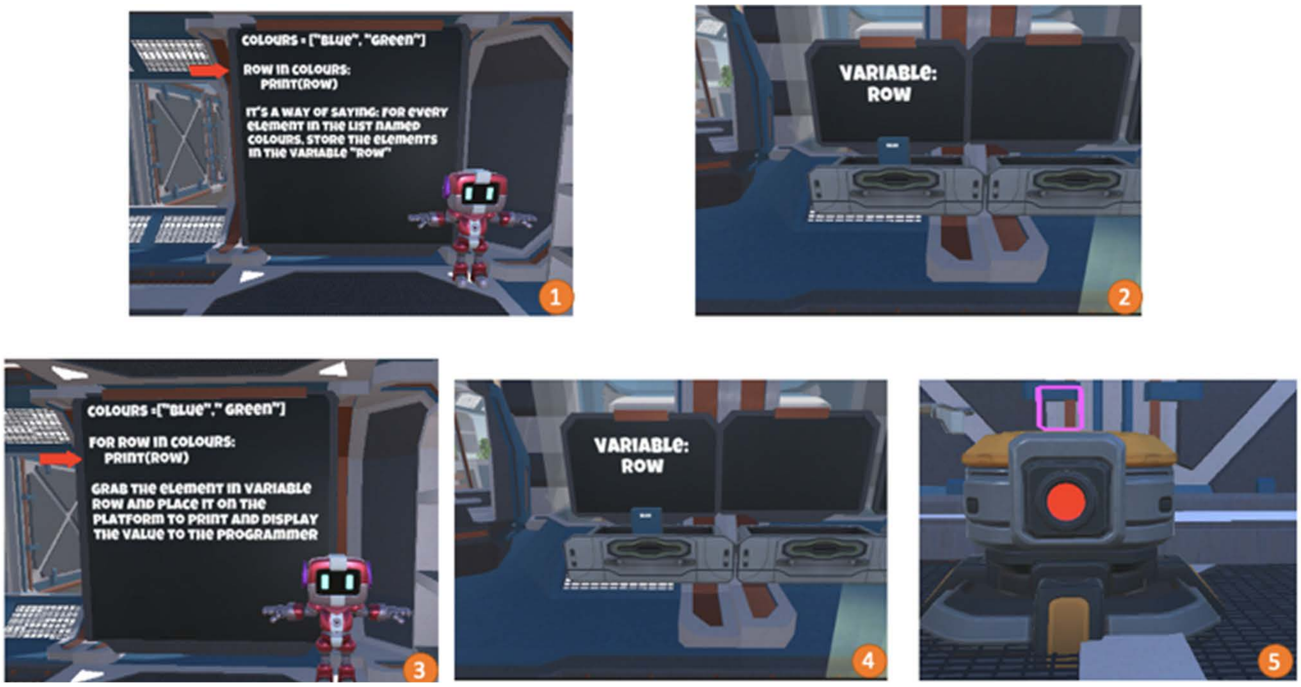
As with most studies, alternative methods are considered when coming up with an approach to address the gaps presented in this study. Some of the alternative methods include visual programming, tangible tools, and game-based learning experiences. However, these techniques are deemed unsuitable for the aim of this study, which is to solve issues faced by students when learning computer programming. For instance, visual programming makes programming less daunting by allowing users to build programs without the need to worry about syntax [51]. However, since this study aims to address misconceptions and since misconceptions of programming concepts also cause syntax errors [21], the nature of visual programming languages that do not expose users to syntax does not coincide with this aim. For the same reason, visual programming does not meet the requirements of the spiral approach which aims to strengthen both semantic and syntactical knowledge at the same time [76]. As mentioned, it is important to have students visualize the computer memory as a large storage space capable of storing many pieces of data [83]. Hence, while tangible tools support the constructivist learning theory [30], [31], it may require extensive effort on the educator's side to prepare elaborate cardboard setups to visualize a computer's memory. However, this is not an issue for pre-developed VR environments that can be shared



```
IF AGE > 20:
    PRINT("YOU ARE TOO OLD!")
```

THE PRINT STATMENT WILL BE RUN IF THE USER IS ABOVE 20 YEARS OLD"

FIGURE 6. (1) An if statement is presented and explained, (2) Block to satisfy the print statement, (3) The platform with a snap zone where the block is supposed to go, to simulate displaying outputs to the programmer.



```
COLOURS = ["BLUE", "GREEN"]
ROW IN COLOURS:
    PRINT(ROW)
```

IT'S A WAY OF SAYING: FOR EVERY ELEMENT IN THE LIST NAMED COLOURS, STORE THE ELEMENTS IN THE VARIABLE "ROW"

```
COLOURS =["BLUE"," GREEN"]
FOR ROW IN COLOURS:
    PRINT(ROW)
```

GRAB THE ELEMENT IN VARIABLE ROW AND PLACE IT ON THE PLATFORM TO PRINT AND DISPLAY THE VALUE TO THE PROGRAMMER

FIGURE 7. (1) A for-loop to traverse through a list is presented and explained, (2) The data "blue" is seen in variable "row" for the first iteration, (3) The print statement is explained, (4) Block to satisfy the print statement, (5) The platform with a snap zone where the block is supposed to go, to simulate displaying outputs to the programmer.

and setup with minimal effort. Lastly, while GBL applications for computer programming has also shown promising results, VR environments may be more engaging than non-VR interventions. This is also supported by the fact that students generally prefer VR interventions over non-VR interventions in educational contexts [95], [96], [97].

IV. EXPERIMENTAL SETUP

The main objective of this research is to determine if iProgVR can help address and mitigate issues faced by students when learning introductory programming. This study will also attempt to compare the effects of the same intervention with and without the presence of VR. The non-VR intervention has all the same aspects of the VR intervention, but players utilize the mouse and the keyboard when carrying out tasks.

A. PARTICIPANTS

To test the feasibility of the system, 60 participants were recruited between ages 18 to 23, of which 30 were female and 30 were male. 30 of the participants indicated that they had previously programmed and identified themselves as beginners while the other 30 indicated that they had never programmed before. 30 participants were presented with the VR intervention (iProgVR) while 30 participants were presented with a video lecture. The video lecture is viewed by students who were currently enrolled in the Programming Principles module at the university as part of their online studies. For simplicity's sake, comparisons done between the experimental and control group will be regarded as Experiment 1 throughout this paper.

To compare the effects of the same intervention with and without the presence of VR, 24 participants were recruited between ages 17 to 22, of which 17 were female and 7 were male. 12 of the participants indicated that they had previously programmed and identified themselves as beginners while the other 12 participants indicated that they had never programmed before. To effectively evaluate the effects of the intervention with and without the presence of VR, 12 participants, 6 of which identified as beginners and 6 without any programming experience were presented with the VR version of the intervention. Similarly, 12 participants, 6 of which identified as beginners and 6 without any programming experience were presented with the non-VR version of the intervention. Again, for simplicity's sake, comparisons done between the group presented with the VR intervention (iProgVR) and the group presented with the non-VR intervention will be regarded as Experiment 2 throughout this paper.

Participants were recruited by posting an advertisement on the school's online management system which students access daily. Those interested will be directed to a form to provide details on when they are available. The sample size was chosen based on the Central Limit Theorem (CLT) which states that when random variables are added, the means of the sample will tend towards a normal distribution regardless of the initial distribution of the population [98]. According to Kwak and Kim, a sample size of 30 is sufficient to hold this theory [99].

B. EXPERIMENTAL PROCEDURE

All experiments conducted for this study employs a two-group pre and post true experimental design to measure participants' programming knowledge, perceived outcomes, and mental workload before and after being presented with the developed VR intervention (iProgVR). A true experiment research design where participants are randomly allocated to experimental and control groups was chosen because it can produce strong comparisons between both groups [100].

Before starting the experiment, participants were briefed on the structure of the study and is made aware that they can drop out of the study if they wish to. The experiment was carried out anonymously, in which participants were not required to reveal their identities. Participants are then given a randomly generated number to decide which group they will be assigned to. Those with odd numbers are assigned to the control group while those with even numbers are assigned to the treatment group. A consent form was also handed out to ensure that their participation was voluntary.

After the consent form was signed, participants (for both experiments) were first given a pre-experimental survey in which they were given 15 minutes to complete. The survey requires participants to answer questions pertaining to their demographics, their perceptions of their programming skill and motivation level. The survey also features questions that were designed specifically to assess their prior understanding of concepts such as variables, lists, 2D arrays, if statements, and loops. When filling up the survey forms, participants were instructed to not discuss their answers with another party to avoid biasness in their answers.

Once the pre-experimental survey is done, participants were then asked to complete a tutorial session to familiarize themselves with the controls and the surrounding environment. Particularly, the tutorial session is carried out to acquaint participants with ways to navigate through the virtual environment, ways to carry out gestures such as grabbing and placing, and the location of important landmarks. These tasks are applicable to both the VR and non-VR intervention. For participants assigned to the control group in experiment 1, they will be presented with slides containing materials discussed in iProgVR. Once it is confirmed that the participant has successfully carried out the tasks in the tutorial session, the participants were then presented with a set of challenges (Table 2).

For experiment 1, once the video and all the challenges in Table 2 were completed by participants in the control and experimental group respectively, participants were then asked to complete the NASA-TLX survey and the post-experimental survey. The NASA-TLX survey is used to access the participant's mental workload while completing tasks in the VR environment. More details regarding this will be provided in Section D below. The post-experimental survey consists of questions that were designed to assess the participant's comprehension of concepts taught during the intervention. Other than that, the post-experimental survey also requires participants to answer questions that assess their perceived outcome. The assessment questions in the post-experimental survey feature questions that were similar to the ones in the pre-experimental questions. This is to accurately measure if there are any improvements before and after being presented with iProgVR. Figure 8 shows the summary of the procedures carried out during experiment 1.

For experiment 2, participants will then be presented with another version of the intervention. For instance, if the participant was first presented with the VR version of the intervention, they will then be presented with the non-VR

TABLE 2. Challenges presented to students with their corresponding goal.

Statement	Goal
total = 10	To reduce abstractness for variables.
total = 10,20	To show that variables do not store more than one value [22], [23], [24].
colours = ["blue", "green"]	To reduce abstractness for lists.
print(colours [0])	To show that the first element of the list is indexed as 0 instead of 1 [28].
temperature = [[11,5], [10,6]]	To reduce abstractness for 2D arrays.
print(temperature[0,0])	To show that the first element of the array is indexed as 0,0 instead of 1,1 [28].
if age > 20: print("You are too old")	To introduce the concept of if statements.
for row in colours: print(row)	To familiarize and introduce the concept of loops in the context of traversing through a list.

for row in temperatures:
    for column in rows:
        print(column)
	To familiarize and introduce the concept of loops in the context of traversing through a 2D array [16].

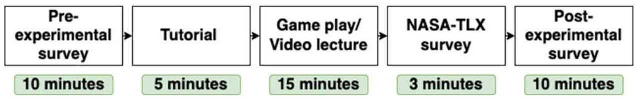


FIGURE 8. Summary of experimental procedures for experiment 1.

version of the intervention and vice versa. The non-VR intervention feature similar content from iProgVR. The only exception being the way participants react with the VR environment. While the iProgVR requires a headset and controllers, the non-VR intervention only requires the standard mouse and keyboard. They will then be asked to complete the NASA-TLX survey again. Figure 9 shows the summary of the procedures carried out during experiment 2.

C. EVALUATION OF PERFORMANCE

The performance of the participants is determined through assessment questions that are present in both pre- and post-experimental surveys. The concept of variables, lists, 2D arrays, if statements, and loops are assessed through 15 questions on each survey, which is divided into 4 sections. For both surveys in the first section (5 questions), participants were asked to describe the concepts mentioned to the best of their ability.

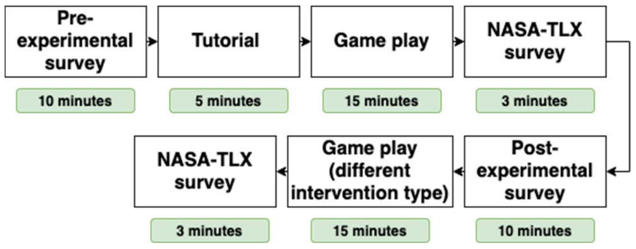


FIGURE 9. Summary of experimental procedures for experiment 2.

TABLE 3. Survey questions to gauge student's perceived outcomes.

No	Questions
Q1	I prefer to have analogies presented to me visually in the virtual environment than spoken about verbally.
Q2	The experience is a good supplement to lessons.
Q3	I prefer learning programming through virtual reality experiences.
Q4	I prefer learning programming through video lectures.
Q5	I now feel more confident about my programming knowledge.
Q6	I found the experience to be engaging.
Q7	I found the concepts presented easy to understand.
Q8	The learning approach enriched the learning process.

TABLE 4. Results of the Shapiro-Wilk test, indicating normality.

Statistic	df	Significance
.972	60	.185

The second section consists of questions specially designed to assess how well the participants were able to apply programming knowledge to the real world. The questions were created based on a book by Briggs [101] and Hayes [102], that specializes in teaching programming to beginners. For example, participants will be asked to identify if the statement can either be represented with a variable, list, array or an if statement.

The third section consists of questions to assess how well participants can apply their knowledge of for-loops by writing code to traverse through lists and 2D arrays. The questions were taken from past year questions from the Programming Principles course from the university. The fourth section consists of questions that are related to common misconceptions. Specifically, these questions are related to the misconception that variables can store more than one value, as supported by by Sleeman *et al.* [22], Doukakis *et al.* [23] and a recent study by Swidan *et al.* [24] and that the first element of an array is

TABLE 5. Paired sample t-test results of participant's performance in the experimental group.

	Mean	Std. Dev	Std. Error	t	df	Sig.
1	-1.3333	.8442	.1541	-8.651	29	<.001
2	-1.167	.791	.145	-8.074	29	<.001
3	-1.5333	.7649	.1396	-10.980	29	<.001
4	-1.1000	.7474	.1365	-8.061	29	<.001
5	-2.1333	2.1453	.3917	-5.447	29	<.001
6	-1.5333	1.8889	.3449	-4.446	29	<.001
7	-1.5333	.8193	.1496	-10.251	29	<.001
8	-1.0833	1.0914	.1993	-5.437	29	<.001

indexed as 1, as supported by Whittall *et al.* [29]. Questions in this section require a true or false answer which is 1 point, as well as an explanation on why the statement is either true or false for another point. Some examples include "To access the first element of a list, the element is referred to by index 1." and "colour = blue, green". The aggregate of all scores gathered from the 4 sections will then be calculated for both pre- and post-surveys.

While the objectives of this study can be achieved by evaluating student performance alone, this study also includes perceived outcomes, and student workload evaluation.

D. EVALUATION OF PERCEIVED OUTCOMES

Questions based on the 7-point Likert Scale to assess participants' perceived outcomes after trying out the system were also included in the post-survey. These questions were designed based on the format of a survey by the Maximizing the Impact of Stem Outreach (MISO) organization [103]. Table 3 shows all the questions included in this part of the questionnaire.

E. WORKLOAD EVALUATION

To assess the workload for the tasks presented to participants, the NASA TLX survey [104] was used. The survey consists of questions to assess mental demand, physical demand, temporal demand, performance, effort, and frustration. Each category is scaled from 0-100, in which 0-9 is considered low, 10-29 is considered medium, 30-49 is considered somewhat high, 50-79 is considered high, and 80-100 is considered very high [104].

F. TESTING ENVIRONMENT

The experiment took place in the House of Multimodal Evolution (HOME) lab at Sunway University, Malaysia for a period of two weeks. The device used for the experiment is a laptop running the Windows 10 Enterprise operating system powered with an Intel Core i7 @ 2.90GHz processor accompanied with 8GB ram. Along with that, the Oculus Rift headset and Oculus Touch controllers were used for participants to experience iProgVR. As for the intervention without VR, participants used a standard keyboard and mouse with the same laptop.

G. DATA ANALYSIS

The paired t-test, the independent t-test, and the ANCOVA test were used to evaluate data gathered from this study. These statistical approaches were chosen as the paired t-test is commonly used to assess participant's pre- and post-scores after being presented with an intervention [105]. Consequently, the independent t-test is commonly used to compare the means of samples from two groups [105]. To analyse student's performance when presented with the developed VR intervention as compared to the video lecture, the ANOVA technique is used as it is often used to analyse the post-test results of two interventions [105]. All tests employed a confidence level of 95%.

Having decided to employ t-tests to analyse data, a normality test (Shapiro-Wilk) was carried out to ensure the samples are normally distributed. This is crucial, as this is an assumption when using t-tests [106]. As shown in Table 4, the test generated a non-significant result (more than.05) which indicates normality [106].

V. RESULTS

A. PERFORMANCE

As mentioned, a paired-samples t-test was carried out to evaluate the impact of iProgVR on participant's scores on the pre- and post-assessment. As shown in Table 5, it was found that the responses to the pre-and post-assessment questions were statistically significant for variables ($M = -1.3333$, $SD = .8442$); lists ($M = -1.167$, $SD = .791$); 2D arrays ($M = -1.5333$, $SD = .7649$); if-statements ($M = -1.1000$, $SD = .7474$); loops ($M = -2.1333, SD = 2.1453$); variable misconception ($M = -1.5333, SD = 1.8889$); list misconception ($M = -1.5333, SD = .8193$); and 2D arrays misconception ($M = -1.0833, SD = 1.0914$); $t(29) = -7.549$, $p =< .05$ for variables; $t(29) = -11.459$, $p =< .0005$ for lists; $t(29) = -10.148$, $p =< .05$ for 2D arrays; $t(29) = -8.061$, $p =< .05$ for if-statements, $t(29) = -5.447$, $p =< .05$ for loops, $t(29) = -4.446$, $p =< .05$ for variable misconception, $t(29) = -10.251$, $p =< .05$ for list misconception, and $t(29) = -5.437$, $p =< .05$ for variable misconception. This signifies that iProgVR successfully increase participant's programming knowledge.

A paired-samples t-test was also carried out to evaluate the impact of the video lecture on participant's scores on the pre- and post-assessment. As shown in Table 6, it was found that the responses to the pre-and post-assessment questions were statistically significant for variables ($M = -53.83$,

TABLE 6. Paired sample t-test results of participant's performance in the control group.

	Mean	Std. Dev	Std. Error	t	df	Sig.
1	-53.83	12.94	2.362	-22.79	29	<.001
2	-2.800	1.031	.1880	-14.88	29	<.001
3	-24.15	7.066	1.290	-18.72	29	<.001
4	-1.3833	.8477	.1548	-8.938	29	<.001
5	-1.9333	1.8229	.3328	-5.809	29	<.001
6	-.9167	1.5148	.2766	-3.315	29	.002
7	-.2333	.8584	.1567	-1.489	29	.147
8	-.1333	.5074	.0926	-1.439	29	.161

SD = 12.94); lists (M = −2.800, SD = 1.031); 2D arrays (M = −24.15, SD = 7.066); if-statements (M = −1.3833, SD =.8477); loops (M = −1.9333, SD = 1.8229) and variable misconception (M = -.9167, SD = 1.5148); t(29) = −22.79, p =< .05 for variables; t(29) = −14.88, p =<.0005 for lists; t(29) = −18.72, p =< .05 for 2D arrays; t(29) = −8.938, p =< .05 for if-statements, t(29) = −5.809, p =< .05 for loops, t(29) = −3.315, and p =< .05 for variable misconception. On the other hand, it was found that the responses to the pre-and post-assessment questions were not statistically significant for list misconception (M = −.9167, SD = 1.5148); and 2D arrays misconception (M = −.1333, SD =.5074); t(29) = −1.489, p => .05 for list misconception, and t(29) = −1.439, p => .05 for variable misconception. This signifies that the video lecture successfully increased participant's programming knowledge to a certain degree.

1: Variables (Pre) - Variables (Post)
2: List (Pre) - List (Post)
3: Arrays (Pre) - Arrays (Post)
4: If-Stat (Pre) - If-Stat (Post)
5: Loops (Pre) - Loops (Post)
6: Variables Misconception (Pre) - Variables Misconception (Post)
7: List Misconception (Pre) - List Misconception (Post)
8: Arrays Misconception (Pre) – Arrays Misconception (Post)

Before carrying out the ANCOVA test on student's performance, the homogeneity of the regression coefficient was first evaluated on the student's pretest scores for both the control and experimental group. It was found that the pretest scores for both groups had a significance level of.113. This indicates that the pre-test scores for both the experimental and control groups are not significantly different, also indicating that

TABLE 7. ANCOVA test results of participant's performance from the experimental and control groups.

	Mean	Std. Dev	Std. Error	N	F	Sig.
1	13.717	4.9562	.9049	30	16.24 8	<.001
2	10.317	5.6332	1.028	30		

TABLE 8. Independent samples t-test results comparing participant's scores according to gender and programming experience.

	Mean	Std. Dev	Std. Error	t	df	Sig.
1	15.033	4.8641	1.2559	1.485	28	.149
2	12.400	4.8484	1.2519			
3	12.467	5.0936	1.3152	-1.404	28	.171
4	14.967	4.6463	1.1997			

the ANCOVA test can be used to analyse student's post test scores. As shown in Table 7, there was a significant difference between both groups on their post-test scores F = 16.248, p <.05. This indicates that iProgVR improved student's scores significantly more than the video lecture.

1: Experimental Group
2: Control Group

An independent-samples t-test was carried out to compare the post-assessment scores for female and male participants. As shown in Table 8, there was no significant difference in scores for female (M = 15.033, SD = 4.8641) and male participants (M = 12.400, SD = 4.8484); t(28) = 1.485, p => .05. This indicates that one's gender does not influence assessment scores. Additionally, it was found that there was no significant difference in scores for beginners (M = 14.967, SD = 4.6463) and those without programming experience (M = 12.467, SD = 5.0936); t(28) = −1.404, p => .05. This indicates that one's past programming experience does not influence assessment scores.

B. PERCEIVED OUTCOME

To assess participant's perceived outcomes, the mean was calculated from the Likert Scale based survey questions. As seen from Table 9, participants in the control group (M = 5.90, SD = 1.062) were more likely to agree that they prefer to have analogies presented to them visually in the virtual environment than spoken about verbally compared to those in the experimental group (M = 5.17, SD = 1.577). It was also found that this is significant, at t(58) = −2.112, p =< .05. Participants in both the experimental (M = 5.70, SD = 1.343) and control group (M = 5.93, SD = 1.202) agree that the iProgVR and the video lecture can be good supplements

TABLE 9. Independent samples t-test results of participant's perceived outcomes from the experimental and control groups.

	G	Mean	Std. Dev	Std. Error	t	df	Sig.
Q1	1	5.17	1.577	.288	-2.112	58	.039
	2	5.90	1.062	.194			
Q2	1	5.70	1.343	.245	-.709	58	.481
	2	5.93	1.202	.219			
Q3	1	5.23	1.357	.248	1.215	58	.229
	2	4.73	1.799	.328			
Q4	1	4.70	1.317	.240	1.095	58	.278
	2	4.27	1.721	.314			
Q5	1	4.40	1.192	.218	-.185	58	.854
	2	4.47	1.570	.287			
Q6	1	5.60	1.133	.207	2.045	58	.045
	2	4.90	1.494	.273			
Q7	1	5.30	1.393	.254	-.095	58	.925
	2	5.33	1.322	.241			
Q8	1	5.83	1.085	.198	1.169	58	.247
	2	5.47	1.332	.243			

to lessons. The difference in scores were not significant, at t(58) = −.709, p => .05.

Participants in both the experimental (M = 5.23, SD = 1.357) and control group (M = 4.73, SD = 1.799) also agree that they prefer to learn programming through VR. The difference in scores were not significant, at t(58)= 1.215, p => .05. Additionally, participants in the experimental group (M = 4.70, SD = 1.317) were more likely to agree that they prefer to learn programming through video lectures compared to participants in the control group (M = 4.27, SD = 1.721). However, this difference in scores were not significant, at t(58) = 1.095, p => .05.

Participants in both the experimental (M = 4.40, SD = 1.192) and control group (M = 4.47, SD = 1.570) expressed neutrality when asked if they feel more confident about their programming knowledge after the experience. As expected, the difference in scores were not significant, at t(58) = −.185, p => .05. When asked if the experience presented were engaging, participants in the experimental group (M = 5.60, SD = 1.133) were more likely to agree compared to participants in the control group (M = 4.90, SD = 1.494). The difference in scores were found to be significant at, t(58) = 2.045, p =< .05. Lastly, when asked if they found the concepts presented easy to understand and if the learning approach enriched the learning process, participants in both the experimental (M = 5.30, SD = 1.393); (M = 5.33, SD = 1.322) and control group (M = 5.83, SD = 1.085); (M = 5.47, SD = 1.332) agreed. The difference in scores for both outcomes were not significant at t(58) = −.095, p => .05 and t(58) = 1.169, p => .05 consecutively.

Q1: I prefer to have analogies presented to me visually in the virtual environment than spoken about verbally.
Q2: The experience is a good supplement to lessons.
Q3: I prefer learning programming through virtual reality experiences.
Q4: I prefer learning programming through video lectures.
Q5: I now feel more confident about my programming knowledge.
Q6: I found the experience to be engaging.
Q7: I found the concepts presented easy to understand.
Q8: The learning approach enriched the learning process.
G1: Experimental Group
G2: Control Group

C. WORKLOAD

Participant's mental, physical, temporal demands as well as performance, effort, and frustration levels were also assessed with the NASA-TLX survey. Table 10 shows the summarised data gathered. It was found that participant's mental load for both the experimental (M = 53.50, SD = 19.080) and control (M = 60.33, SD = 20.040) group was high. It was also found that participant's physical load, temporal load for both the experimental (M = 41.33, SD = 23.887); (M = 39.67, SD = 20.675) and control (M = 36.17, SD= 28.122); (M = 44.00, SD = 25.977) group was somewhat high. Additionally, participants from both the experimental (M = 48.50, SD = 24.570); (M = 50.33, SD = 20.924); (M = 37.33, SD = 23.183) and control (M = 56.67, SD = 21.267); (M = 60.17, SD = 20.531); (M = 39.17, SD = 31.815) group rated their performance, effort levels to be high and their frustration levels to be somewhat high. For mental (t(58) = −1.353, p => .05), physical (t(58) = −.767, p => .05), and temporal load (t(58)= −.715, p => .05), the difference in scores between both groups were found to be not significant. Consequently, the difference in scores for level of performance (t(58) = −1.376, p => .05), effort (t(58) = −1.837, p => .05), and frustration (t(58) = −.255, p => .05) was also found to be not significant for both groups.

W1: Mental Demand
W2: Physical Demand
W3: Temporal Demand
W4: Performance
W5: Effort
W6: Frustration
G1: Experimental Group
G2: Control Group

D. PRELIMINARY RESULTS (EXPERIMENT 2)

An independent samples t-test was carried out to determine if there are significant differences between the performance of participants who were presented with the VR and non-VR

TABLE 10. Independent samples t-test results of participant's perceived outcomes from the experimental and control groups.

	G	Mean	Std. Dev	Std. Error	t	df	Sig.
W1	1	53.50	19.080	3.484	-1.353	58	.181
	2	60.33	20.040	3.659			
W2	1	41.33	23.887	4.361	-.767	58	.446
	2	36.17	28.122	5.134			
W3	1	39.67	20.675	3.775	-.715	58	.478
	2	44.00	25.977	4.743			
W4	1	48.50	24.570	4.486	-1.376	58	.174
	2	56.67	21.267	3.883			
W5	1	50.33	20.924	3.820	-1.837	58	.071
	2	60.17	20.531	3.748			
W6	1	37.33	23.183	4.233	-.255	58	.800
	2	39.17	31.815	5.809			

TABLE 11. Independent samples t-test results of participant's performance from the experimental and control groups.

	Mean	Std. Dev	Std. Error	t	df	Sig.
1	6.875	1.9203	.5543	-.337	22	.739
2	7.167	2.2995	.6638			

1: Experimental Group (VR intervention)

2: Control Group (Non-VR version of the same intervention)

version of iProgVR. As shown in Table 11, there was not a significant difference in scores for the control (M = 7.167, SD = 2.2995) and experimental group (M = 6.875, SD = 1.9203); t(22) = −.337, p => .05. This indicates that participant's assessment scores are not influenced by the presence or absence of VR.

1: Experimental Group (VR intervention)

2: Control Group (Non-VR version of the same intervention)

VI. DISCUSSION

A. PERFORMANCE

As seen in Table 5, students showed significant improvement in their performance post- intervention for all concepts presented. This suggests that by representing abstract concepts as concrete objects and by allowing students to identify and rectify misconceptions, programming knowledge can be improved. Despite so, performance scores were rather low at the end of the evaluation. This is expected, given the short amount of time allowed to learn four major programming concepts as well as programming misconceptions.

The first research question posed was "How will the representation of programming concepts as concrete objects in the virtual environment aid in students' understanding of abstract concepts?" Along with this research question, the hypothesis proposed states that "Abstract programming concepts are easier to understand if it is represented as concrete objects in the virtual environment." The hypothesis is supported by the evaluation results, whereby students' post-intervention scores for the variables, arrays, if statements, and loops were significantly better than the pre-intervention scores. This suggests that by representing abstract concepts as concrete objects in the virtual environment, programming knowledge can be improved. These results build on existing evidence of the ability of VR to help students effectively visualize abstract programming concepts as seen in [34].

The second research question posed was "How will the ability to identify and address common misconceptions of programming concepts by interacting with the virtual environment aid in reducing misconceptions?" Along with this research question, the hypothesis proposed states that "By allowing students to interact with the virtual environment, programming misconceptions can be better identified and addressed." As seen in Table 5, the hypothesis is supported by the evaluation results, whereby students' post-intervention scores for the variable, list, and 2D array-related misconception were significantly better than the pre-intervention scores. This suggests that by allowing students to identify and address programming misconceptions in the virtual environment, misconceptions can be mitigated. These results build on existing evidence of the being able to improve knowledge acquisition when allowed to interact with one's environment [107], [155], [156].

Furthermore, as seen in Table 6, comparisons in performance scores between the VR intervention (iProgVR) and the video lecture was significant. This indicates that participants who were presented with the iProgVR scored significantly better than those who were presented with the video lecture. As online classes become more popular due to the Covid-19 pandemic, this finding can be useful for future applications of remote learning. This finding is synonymous with findings from existing studies that utilised VR and traditional teaching methods in comparing student's assessment scores [39], [108]. However, as seen in Table 10, the difference in performance scores for both the VR and non-VR intervention was not significant. While previous research has focused on comparing their developed VR intervention with popular non-VR programming boot camps like Kodu and Blockly [36], [39], there is a relative paucity of studies that compared the VR and non-VR version of the same intervention in the context of teaching and learning computer programming. Hence, this result demonstrates that the presence or absence of VR does not affect knowledge attainment. However, these results should not be used as a basis because there is a clear limitation when it comes to the number of sample size in experiment 2. It is important to note that this experiment is preliminary and is meant to provide a foundation for future research.

Comparing post-assessment scores between beginners and those without any programming experience, performance scores were not significant as seen in Table 7. This shows that the system is suitable for those with or without programming experience. These results build on existing evidence of students, irrespective of programming experience performing well after being presented with a VR intervention for learning computer programming [39]. Comparisons between female and male participants post-assessment scores was also not significant. Current works have mixed findings when investigating how different genders perform in the context of serious games. For instance, when it comes to tasks that require situation analysis and abstract thinking, male students generally outperform female students in this aspect [109], [110]. However, with regards to attitude, female students generally reported higher perceptions of positive affective quality compared to male students [111]. Hence, further research is required to establish how different genders perform or perceive VR interventions, especially those developed specifically for computer programming learning.

B. PERCEIVED OUTCOME

Furthermore, participants in both groups agree that they prefer to have analogies presented to them in VR than spoken about in class verbally. While previous research has shown that the usage of analogies in the teaching of computer programming can help in increasing knowledge acquisition [65], [34], these results demonstrate that students prefer to learn by visualizing abstract concepts in VR. This can be used as a basis for future research when determining the significance of VR in helping students visualize abstract concepts. Results also show that participants (both groups) agree that the experience was a good supplement to conventional lessons. This is corroborated by previous studies which has shown that VR can help supplement lessons [112], [113].

When asked if they prefer learning through VR experiences, those presented with the VR intervention were more likely to agree compared to those presented with the video lecture. While previous research has found that students generally prefer VR over a non-VR setting when learning [95], this result provides a new insight into the relationship between exposure and interest to learn with VR. These results also build on existing evidence of students generally preferring VR over a non-VR setting in educational contexts [95], [96], [97]. Surprisingly, when asked if they preferred learning through video lectures, participants in both groups also agreed. This indicates that students are mostly interested to experience new teaching forms. Results from this study can be used as a basis for improving remote learning experiences as this way of learning becomes more popular. In terms of perceived confidence levels, participants (both groups) agree that the experience increased their confidence in terms of programming knowledge. While existing evidence exists on the ability of VR to improve confidence levels [114], this indicates that while VR interventions can increase one's perceived confidence on a particular subject matter, video lectures are no different.

The third research question posed was, "Will the developed intervention increase student engagement when learning computer programming?" Along with this research question, the hypothesis proposed states that "The developed intervention can help with student motivation and engagement when learning computer programming." As seen in Table 8 (Q6), the hypothesis is supported by the evaluation results, whereby participants who were presented with the iProgVR reported significantly higher engagement scores compared to participants who were presented with the video lecture. These results build on existing VR and GBL interventions for computer programming learning that reported high levels of student engagement [38], [39].

Lastly, when asked if the concepts presented was easy to understand, participants in both groups agreed. This indicates that they found that the concepts were presented in a simple and easy way. Interestingly, this also means that the mere existence of VR does not make content less complicated. Which indicates that if lessons are presented appropriately, knowledge can be absorbed by students efficiently with or without the presence of VR. This is further supported by participants (both groups) agreeing that both VR and video lectures can enrich the learning process. Despite so, participants who were presented with the iProgVR had better assessment scores than participants who learnt through video lectures. This is a weird discrepancy as participants equally agree that both experiences made learning easy and enriched learning. One explanation that can be given is that while the video lecture did a good job in presenting the concepts, student's may have been bored by the monotonous nature of the medium, thereby decreasing retention rates. This can be corroborated by comments given by one participant who said: "The presentation can be more engaging. The slide itself was not eye-catching and I believe there a more concise methods used by YouTube tutors which I find to be more helpful in terms of ever-lasting knowledge. Besides that, this lecture was helpful."

C. WORKLOAD

As seen in Table 9, participants (both groups) expressed that the mental demand needed to complete both interventions was high. This indicates that both interventions required participants to use rather high levels of thinking, calculating, searching, and remembering when completing challenges presented to them. This is expected as participants had to learn similar concepts in both interventions. These results are synonymous with findings found in an article by Berkman *et al.* [115] who reported that comparisons of mental demand scores in VR and without VR for a puzzle game were not significant. However, there are also conflicting findings that report significantly higher mental demand scores for the VR version of the experience compared to the non-VR version of the same experience [116], [117], [118].

Participants also rated the physical demand to be somewhat high for the VR intervention (iProgVR). This could be due to the need for physical activities such as turning one's head and body to teleport around the VR environment. The need to grasp virtual objects with the use of controllers could also be the reason. Surprisingly, participants also rated physical demand to be somewhat high for the video lecture as well. One potential reason could be the need to remain stationary while watching the video. These results are synonymous with findings found in an article by Rao *et al.* [116] who reported that comparisons of physical demand scores in VR and without VR for a military training simulator were not significant. However, Berkman *et al.* [115] reported significantly higher physical demand scores for the VR version of their puzzle game compared to the non-VR version of the same game.

Furthermore, in terms of temporal demand, participants also rated it to be somewhat high for both interventions. This indicates that participants felt some time pressure due to the rate or pace at which the task occurred. The time set for participants to experience both interventions was set to 15 minutes. While all participants were able to carry out all the tasks in the allotted time, the explicit mention of a time limit could be the reason why temporal demand was rated somewhat high. These results are synonymous with findings found in an article by Berkman *et al.* [115] who reported that comparisons of temporal demand scores in VR and without VR for a puzzle game were not significant. However, Rao *et al.* [116] reported significantly higher temporal demand scores for the VR version of their military training simulator compared to the non-VR version of the same simulator.

Results showed that participants in the experimental and control group rated their ability to accomplish the goal of the tasks and how satisfied they were about their performance to be somewhat high and high consecutively. However, the difference in performance scores for both interventions were not significant. This indicates that while the tasks required some degree of mental, physical, and temporal demand, participants were still able to complete the tasks and feel satisfied with their accomplishments. These results are synonymous with findings found in an article by Berkman *et al.* [115] and Rao *et al.* [116].

When asked about how much mental and physical effort was needed to accomplish the tasks presented to them, participants (both groups) rated it to be high. As mentioned, one possible reason why the physical effort was rated somewhat high is the need to physically use certain parts of the body to navigate and interact with the environment in the VR intervention. Consequently, the need to process new information on the spot to correctly accomplish tasks or finishing up the post-survey could be mentally demanding for participants. These results are synonymous with findings found in an article by Berkman *et al.* [115] and Rao *et al.* [116].

Participants were also asked about their frustration level, particularly how stressed, annoyed, and discouraged did they feel during the tasks. It was found that participants in both the experimental and control group rated their frustration level to be high. This could also be due to the need to learn so many new concepts, one after the other in a short period of time. These results are synonymous with findings found in an article by Rao *et al.* [116]. However, there are also conflicting findings that report significantly higher frustration levels for the VR version [117] while another study reported report significantly higher frustration levels for the non-VR version of iProgVR [115].

The inconsistencies in results from this study and existing works when it comes to the workload of VR games and interventions indicate that workload is largely dependent on the tasks one needs to carry out in the virtual environment. For instance, the workload may differ across VR and non-VR platforms for a puzzle game [115], a shoot training simulator [116], [117], and a wheelchair simulator [118]. Regardless, the workload results discussed above can be a good benchmark for future research when determining workloads for VR and video lectures in the context of teaching and learning computer programming.

D. FEEDBACK

This section highlights participants' feedback that ranges from ways to improve the system and the type of intervention preferred. Listed below are some of the comments left by participants:

1. "I would like to have the ability to interact with others in the virtual environment."
2. "I would like to walk around the virtual environment."
3. "I would like to experience the VR intervention wirelessly."
4. "I would like a more appealing colour scheme for the virtual content."
5. "A workshop featuring this intervention should be organized."
6. "I would like more time to learn the concepts introduced to me."

As seen above, feedbacks left by participants are useful for future renditions of the intervention. For instance, the current work can be expanded to feature collaborative elements. Thus, more tests should be done to find out if performance can be increased when participants are allowed to work and interact with one another in the virtual environment. Participants also expressed interest for more immersive environments, particularly one where they can explore by walking around. Related to this, is the desire to move freely, without the restriction of wires that are connected to the VR headset. While this can be achieved with wireless VR headsets like the Oculus Quest, an expensive contraption such as the KAT Walk C would need to be purchased to simulate walking around the virtual environment. Participants also provided suggestions to change the colour scheme of the virtual environment into less drabby and brighter colours. One participant also expressed their interest for a workshop that feature iProgVR. This can certainly be done once the pandemic situation improves. Lastly, some participants also wished to have more time to process the concepts that were presented to them in the virtual environment. As mentioned, this would greatly improve performance.

After completing the post survey, participants in the experimental group were presented with the video lecture while participants in the control group was presented with the VR intervention (iProgVR). This is to determine the preference of every participant. It was found that out of 60 participants, 10 favoured the video lecture while 6 were neutral. 1 participant did not provide an answer to which intervention was preferred. Participants cite reasons such as VR motion sickness and familiarity as to why they prefer the video lecture. For participants who prefer the VR intervention (iProgVR), the most frequently cited reasons range from the intervention being fun, interesting, and easy to understand.

E. POTENTIAL APPLICATIONS

Seeing that this study has garnered positive outcomes, this section aims to list down some potential applications of the developed framework. Firstly, the developed framework can aid in engineering and physics education. More specifically, it can be applied to develop engineering modules for complex and abstract concepts such as electromagnetism and thermodynamics. For example, a viable analogy to explain electromagnetism is by explaining that a ball on a rotating turntable produces orbits that are like particles in a magnetic field [119]. Hence, by adapting this framework, students can visualize and understand electromagnetism easily.

Furthermore, this framework can be adapted to teach calculus. As proposed by the Mathematics Faculty at the University of Cambridge, a curve can be analogised as a road on a map [120]. Hence, by adapting this framework, students can visualize this if presented with a VR experience that simulates driving along the road in conjunction with how it relates to a mathematical curve.

Another abstract computer programming concept that can benefit from the developed framework is recursion. For example, the most popular analogy used to explain recursion is by using Russian dolls. By comparing and visualizing the Russian doll to a call stack (which is an integral element of the recursion algorithm), students can more easily understand the concept. By visualizing the concept in a 3-dimensional space, it will also address any misconceptions students have about recursion.

Other than abstract scientific concepts, the developed model can also be potentially used to visualize abstract emotions such as grief and death. According to Botella *et al.* [121], VR allows individuals to process loss in a physical way. Hence, by applying the framework and representing the loss as a virtual object in the computer-generated space, it is hypothesised that grief can be more effectively tended to.

With the increasing popularization of VR technology, it is not a surprise if virtual environments are adapted more in future classrooms. Results from this study may also be applied to provide a foundation to make VR classrooms a reality.

VII. 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 is proposed. Thirdly, is the need to incite intrinsic motivation when learning programming. To handle this issue, a simulation technique to incite learning motivation and a simulation module is also proposed. Results gathered from testing indicated that the intervention successfully improved student's knowledge acquisition as seen from the significant improvements in the pre- to post-test scores. Furthermore, results also show that students were motivated and engaged throughout the whole experience. This indicates that unlike existing tools for teaching computer programming (as discussed in Section II), the intervention developed can be used as a one stop solution to mitigate issues commonly faced by students when learning programming. Results also showed that there is no difference when it comes to workload and perceived outcomes (except engagement) when comparing iProgVR and the video lecture. However, it was found that students were more engaged and scored higher assessment scores in the experimental group. This means that iProgVR is a feasible system. Comparisons were also done with the VR and non-VR version of the same intervention to address the lack of such studies, specifically in the context of computer programming education. It was found that the presence of VR does not affect knowledge attainment. However, since the nature of experiment 2 is at its preliminary stage, these results should not be used as a basis. As mentioned, since most works in this field are limited to conference papers [90], this means that not much comprehensive testing has been done in existing works. To address this, this study has provided new knowledge by conducting more thorough test that includes workload, pre- and post-assessments of both experimental and control groups, and perceived outcomes that are measuring more than student engagement and motivation (as seen in Table 1). Additionally, since there is also a relative paucity of studies that compared the VR and non-VR version of the same intervention in the context of teaching and learning computer programming, this study would also aim to carry out such comparisons albeit in a preliminary manner, to provide a foundation for future studies.

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, concepts such as loops, or recursion should also be introduced with the system. More syntax and semantic-based misconceptions should also be implemented to address common programming misconceptions. While this study is meant to benefit instructors who are already aware of the benefits of VR and is looking for VR experiences that can do more than engage students when learning programming, future work can also be done to assess iProgVR to other active learning methods. 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 proposed framework can also be applied for use to teach abstract concepts in other domains such as engineering, science, and mathematics and even grief management.

REFERENCES

[1] S. Fayer, A. Lacey, and A. Watson, "STEM occupations: Past, present, and future," U.S. Bureau Labor Statist., Washington, DC, USA, Tech. Rep., 2017. [Online]. Available: https://www.bls.gov/spotlight/2017/science-technology-engineering-and-mathematics-stem-occupations-past-present-and-future/pdf/science-technology-engineering-and-mathematics-stem-occupations-past-present-and-future.pdf

[2] *Higher Education Student Statistics: U.K., 2018/19*, Higher Educ. Statist. Agency, Cheltenham, U.K., 2020.

[3] *Non-Continuation: U.K. Performance Indicators 2018/19*, Higher Educ. Student Statist., Cheltenham, U.K., 2020.

[4] M. N. Giannakos, T. Aalberg, M. Divitini, L. Jaccheri, P. Mikalef, I. O. Pappas, and G. Sindre, "Identifying dropout factors in information technology education: A case study," in *Proc. IEEE Global Eng. Educ. Conf. (EDUCON)*, Athens, Greece, Apr. 2017, pp. 1187–1194, doi: 10.1109/EDUCON.2017.7942999.

[5] I. O. Pappas, M. N. Giannakos, and L. Jaccheri, "Investigating factors influencing Students' intention to dropout computer science studies," in *Proc. ACM Conf. Innov. Technol. Comput. Sci. Educ.*, Arequipa, Peru, Jul. 2016, pp. 198–203, doi: 10.1145/2899415.2899455.

[6] P.-H. Tan, C.-Y. Ting, and S.-W. Ling, "Learning difficulties in programming courses: Undergraduates' perspective and perception," in *Proc. Int. Conf. Comput. Technol. Develop.*, Kota Kinabalu, Malaysia, 2009, pp. 42–46, doi: 10.1109/ICCTD.2009.188.

[7] R. P. Medeiros, G. L. Ramalho, and T. P. Falcão, "A systematic literature review on teaching and learning introductory programming in higher education," *IEEE Trans. Educ.*, vol. 62, no. 2, pp. 77–90, May 2019, doi: 10.1109/TE.2018.2864133.

[8] C. Wee and K. M. Yap, "Design and analysis of a virtual reality game to address issues in introductory programming learning," in *Intelligent Technologies for Interactive Entertainment*, vol. 377, N. Shaghaghi, F. Lamberti, B. Beams, R. Shariatmadari, and A. Amer, Eds. Cham, Switzerland: Springer, 2021, pp. 243–254, doi: 10.1007/978-3-030-76426-5_16.

[9] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming: 12 years later," *ACM Inroads*, vol. 10, no. 2, pp. 30–36, 2019, doi: 10.1145/3324888.

[10] B. A. Becker, C. Murray, T. Tao, C. Song, R. McCartney, and K. Sanders, "Fix the first, ignore the rest: Dealing with multiple compiler error messages," in *Proc. 49th ACM Tech. Symp. Comput. Sci. Educ. (SIGCSE)*, Baltimore, MD, USA, 2018, pp. 634–639. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3159450.3159453

[11] E. Dunican, "Making the analogy: Alternative delivery techniques for first year programming courses," in *Proc. 14th Workshop Psychol. Program. Interest Group*, 2002, pp. 89–99.

[12] S. Dasuki and A. Quaye, "Undergraduate Students' failure in programming courses in institutions of higher education in developing countries: A Nigerian perspective," *Electron. J. Inf. Syst. Developing Countries*, vol. 76, no. 1, pp. 1–18, Sep. 2016, doi: 10.1002/j.1681-4835.2016.tb00559.x.

[13] L. M. M. Giraffa, M. C. Moraes, and L. Uden, "Teaching object-oriented programming in first-year undergraduate courses supported by virtual classrooms," in *Proc. 2nd Int. Workshop Learn. Technol. Educ. Cloud*, L. Uden, Y.-H. Tao, H.-C. Yang, and I.-H. Ting, Eds. Dordrecht, The Netherlands: Springer, 2014, pp. 15–26, doi: 10.1007/978-94-007-7308-0_2.

[14] A. Gomes and A. Mendes, "A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Madrid, Spain, Oct. 2014, pp. 1–8, doi: 10.1109/FIE.2014.7044086.

[15] N. Eltegani and L. Butgereit, "Attributes of students engagement in fundamental programming learning," in *Proc. Int. Conf. Comput., Control, Netw., Electron. Embedded Syst. Eng. (ICCNEEE)*, Khartoum, Sudan, Sep. 2015, pp. 101–106, doi: 10.1109/ICCNEEE.2015.7381438.

[16] S. Xinogalos, "Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy," *Educ. Inf. Technol.*, vol. 21, no. 3, pp. 559–588, May 2016, doi: 10.1007/s10639-014-9341-9.

[17] H. Aris, "Improving students performance in introductory programming subject: A case study," in *Proc. 10th Int. Conf. Comput. Sci. Educ. (ICCSE)*, Cambridge, U.K., Jul. 2015, pp. 657–662, doi: 10.1109/ICCSE.2015.7250328.

[18] M. Rizvi and T. Humphries, "A scratch-based CS0 course for at-risk computer science majors," in *Proc. Frontiers Educ. Conf.*, Seattle, WA, USA, Oct. 2012, pp. 1–5, doi: 10.1109/FIE.2012.6462491.

[19] T. Koulouri, S. Lauria, and R. D. Macredie, "Teaching introductory programming: A quantitative evaluation of different approaches," *ACM Trans. Comput. Educ.*, vol. 14, no. 4, pp. 1–28, Feb. 2015, doi: 10.1145/2662412.

[20] Y. Qian and J. Lehman, "Students' misconceptions and other difficulties in introductory programming: A literature review," *ACM Trans. Comput. Educ.*, vol. 18, no. 1, pp. 1–24, Dec. 2017, doi: 10.1145/3077618.

[21] T. Kohn, "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*, Minneapolis, MN, USA, 2019, pp. 524–530. [Online]. Available: https://dl.acm.org/doi/10.1145/3287324.3287381

[22] D. Sleeman, R. T. Putnam, J. Baxter, and L. Kuspa, "Pascal and high school students: A study of errors," *J. Educ. Comput. Res.*, vol. 2, no. 1, pp. 5–23, Feb. 1986, doi: 10.2190/2XPP-LTYH-98NQ-BU77.

[23] D. Doukakis, M. Grigoriadou, and G. Tsaganou, "Understanding the programming variable concept with animated interactive analogies," in *Proc. 8th Hellenic Eur. Res. Comput. Math. Appl. Conf.*, 2007, pp. 1–8.

[24] A. Swidan, F. Hermans, and M. Smit, "Programming misconceptions for school students," in *Proc. ACM Conf. Int. Comput. Educ. Res.*, Espoo, Finland, Aug. 2018, pp. 151–159, doi: 10.1145/3230977.3230995.

[25] L. Ma, "Investigating and improving novice programmers' mental models of programming concepts," Ph.D. dissertation, Univ. Strathclyde, Glasgow, U.K., 2007.

[26] T. Sirkiä and J. Sorva, "Exploring programming misconceptions: An analysis of Student mistakes in visual program simulation exercises," in *Proc. 12th Koli Calling Int. Conf. Comput. Educ. Res.*, Koli, Finland, 2012, pp. 19–28, doi: 10.1145/2401796.2401799.

[27] T. Kohn, "Variable evaluation: An exploration of novice programmers' understanding and common misconceptions," in *Proc. ACM SIGCSE Tech. Symp. Comput. Sci. Educ.*, Seattle, WA, USA, Mar. 2017, pp. 345–350, doi: 10.1145/3017680.3017724.

[28] L. Chiodini, I. M. Santos, A. Gallidabino, A. Tafliovich, A. L. Santos, and M. Hauswirth, "A curated inventory of programming language misconceptions," in *Proc. 26th ACM Conf. Innov. Technol. Comput. Sci. Educ.*, Jun. 2021, pp. 380–386, doi: 10.1145/3430665.3456343.

[29] S. J. Whittall, W. A. C. Prashandi, G. L. S. Himasha, D. I. De Silva, and T. K. Suriyawansa, "CodeMage: Educational programming environment for beginners," in *Proc. 9th Int. Conf. Knowl. Smart Technol. (KST)*, Chonburi, Thailand, Feb. 2017, pp. 311–316, doi: 10.1109/KST.2017.7886101.

[30] J. C. Chwen, "Theoretical bases for using virtual reality in education," *Themes Sci. Technol. Educ.*, vol. 2, pp. 71–90, Oct. 2009.

[31] E. Fokides and C. Tsolakidis, "Virtual reality in education: A theoretical approach for road safety training to students," *Eur. J. Open, Distance E-Learn.*, vol. 2, no. 11, pp. 1–7, 2008.
[32] T. Leung, F. Zulkernine, and H. Isah, "The use of virtual reality in enhancing interdisciplinary research and education," 2018, *arXiv:1809.08585*.
[33] M. Cowart, "Embodied cognition," *Internet Encyclopedia Philosophy*, vol. 4, no. 3, pp. 319–325, 2018.
[34] T. Tanielu, R. 'Akau'ola, E. Varoy, and N. Giacaman, "Combining analogies and virtual reality for active and visual object-oriented programming," in *Proc. ACM Conf. Global Comput. Educ.*, Chengdu, China, May 2019, pp. 92–98, doi: 10.1145/3300115.3309513.
[35] G. Singh, "Using virtual reality for scaffolding computer programming learning," in *Proc. 23rd ACM Symp. Virtual Reality Softw. Technol.*, Gothenburg, Sweden, Nov. 2017, pp. 1–2, doi: 10.1145/3139131.3141225.
[36] J. Vincur, M. Konopka, J. Tvarozek, M. Hoang, and P. Navrat, "Cubely: Virtual reality block-based programming environment," in *Proc. 23rd ACM Symp. Virtual Reality Softw. Technol.*, 2017, pp. 1–2.
[37] N. Bouali, E. Nygren, S. S. Oyelere, J. Suhonen, and V. Cavalli-Sforza, "Imikode: A VR game to introduce OOP concepts," in *Proc. 19th Koli Calling Int. Conf. Comput. Educ. Res.*, Koli, Finland, Nov. 2019, pp. 1–2, doi: 10.1145/3364510.3366149.
[38] J. Chen, M. R. Zargham, M. Rajendren, and J. Cheng, "Coding VR games," in *Proc. Int. Conf. Frontiers Educ., CS CE*, 2019, pp. 123–127.
[39] R. J. Segura, F. J. Pino, C. J. Ogáyar, and A. J. Rueda, "VR-OCKS: A virtual reality game for learning the basic concepts of programming," *Comput. Appl. Eng. Educ.*, vol. 28, no. 1, pp. 31–41, Jan. 2020, doi: 10.1002/cae.22172.
[40] J. Stigall and S. Sharma, "Virtual reality instructional modules for introductory programming courses," in *Proc. IEEE Integr. STEM Educ. Conf. (ISEC)*, Princeton, NJ, USA, Mar. 2017, pp. 34–42, doi: 10.1109/ISECon.2017.7910245.
[41] D. Kao, C. Mousas, A. J. Magana, D. F. Harrell, R. Ratan, E. F. Melcer, B. Sherrick, P. Parsons, and D. A. Gusev, "Hack.VR: A programming game in virtual reality," 2020, *arXiv:2007.04495*.
[42] Q. Jin, Y. Liu, Y. Yuan, L. Yarosh, and E. Rosenburg, "VWorld: An immersive VR system for learning programming," in *Proc. Interact. Design Children (IDC)*, London, U.K., Jun. 2020, pp. 235–240.
[43] M. R. Zargham and B. Kamsani, "Programming in a virtual reality environment," in *Proc. Int. Conf. Frontiers Educ., Comput. Sci. Comput. Eng.*, Las Vegas, NV, USA, Jul. 2016, pp. 100–103.
[44] D. Parmar, J. Bertrand, S. V. Babu, K. Madathil, M. Zelaya, T. Wang, J. Wagner, A. K. Gramopadhye, and K. Frady, "A comparative evaluation of viewing metaphors on psychophysical skills education in an interactive virtual environment," *Virtual Reality*, vol. 20, no. 3, pp. 141–157, Sep. 2016, doi: 10.1007/s10055-016-0287-7.
[45] A. Sabuncuoğlu, M. Erkaya, O. T. Buruk, and T. Göksun, "Code notes: Designing a low-cost tangible coding tool for/with children," in *Proc. 17th ACM Conf. Interact. Design Children*, Trondheim Norway, Jun. 2018, pp. 644–649, doi: 10.1145/3202185.3210791.
[46] Y. A. C. González and A. G.-V. Muñoz-Repiso, "A robotics-based approach to foster programming skills and computational thinking: Pilot experience in the classroom of early childhood education," in *Proc. 6th Int. Conf. Technol. Ecosyst. Enhancing Multiculturality*, Salamanca, Spain, Oct. 2018, pp. 41–45, doi: 10.1145/3284179.3284188.
[47] S. Lerner, "Projection boxes: On-the-fly reconfigurable visualization for live programming," in *Proc. Conf. Hum. Factors Comput. Syst.*, Honolulu, HI, USA, Apr. 2020, pp. 1–7, doi: 10.1145/3313831.3376494.
[48] M. Mladenović, Ž. Žanko, and M. A. Čuvič, "The impact of using program visualization techniques on learning basic programming concepts at the K–12 level," *Comput. Appl. Eng. Educ.*, vol. 29, no. 1, pp. 145–159, Jan. 2021, doi: 10.1002/cae.22315.
[49] P. Khaloo, M. Maghoumi, E. Taranta, D. Bettner, and J. Laviola, "Code park: A new 3D code visualization tool," in *Proc. IEEE Work. Conf. Softw. Vis. (VISSOFT)*, Shanghai, China, Sep. 2017, pp. 43–53, doi: 10.1109/VISSOFT.2017.10.
[50] B. Jost, M. Ketterl, R. Budde, and T. Leimbach, "Graphical programming environments for educational robots: Open Roberta—Yet another one?" in *Proc. IEEE Int. Symp. Multimedia*, Taiwan, Dec. 2014, pp. 381–386, doi: 10.1109/ISM.2014.24.
[51] X.-J. Bai and B.-L. Liu, "Application of visual programming in program design course," *DEStech Trans. Social Sci., Educ. Hum. Sci.*, 2019, doi: 10.12783/dtssehs/eiem2018/26901.
[52] F. Rahman, "Leveraging visual programming language and collaborative learning to broaden participation in computer science," in *Proc. 19th Annu. SIG Conf. Inf. Technol. Educ.*, Fort Lauderdale, FL, USA, 2018, pp. 172–177, doi: 10.1145/3241815.3242586.
[53] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: programming for all," *Commun. ACM*, vol. 52, no. 11, p. 60, Nov. 2009, doi: 10.1145/1592761.1592779.
[54] S. Cooper, W. Dann, and R. Pausch, "Alice: A 3-D tool for introductory programming concepts," *J. Comput. Sci. Colleges*, vol. 15, no. 5, pp. 107–116, 2000.
[55] E. Pasternak, R. Fenichel, and A. N. Marshall, "Tips for creating a block language with blockly," in *Proc. IEEE Blocks Beyond Workshop (B B)*, Raleigh, NC, USA, Oct. 2017, pp. 21–24, doi: 10.1109/BLOCKS.2017.8120404.
[56] M. Puckette, "Pure data: Another integrated computer music environment," in *Proc. 2nd Intercollege Comput. Music Concerts*, 1996, pp. 37–41.
[57] A. Fowler and B. Cusack, "Kodu game lab: Improving the motivation for learning programming concepts," in *Proc. 6th Int. Conf. Found. Digit. Games*, Bordeaux, France, 2011, pp. 238–240, doi: 10.1145/2159365.2159398.
[58] H. Tsukamoto, Y. Takemura, Y. Oomori, I. Ikeda, H. Nagumo, A. Monden, and K.-I. Matsumoto, "Textual vs. Visual programming languages in programming education for primary schoolchildren," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Erie, PA, USA, Oct. 2016, pp. 1–7, doi: 10.1109/FIE.2016.7757571.
[59] M. Seraj, C. S. Grosse, S. Autexier, and R. Drechsler, "Look what i can do: Acquisition of programming skills in the context of living labs," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Softw. Eng. Educ. Training (ICSE-SEET)*, Montreal, QC, Canada, May 2019, pp. 197–207, doi: 10.1109/ICSE-SEET.2019.00029.
[60] A. Rao, A. Bihani, and M. Nair, "Milo: A visual programming environment for data science education," in *Proc. IEEE Symp. Vis. Lang. Hum.-Centric Comput. (VL/HCC)*, Lisbon, Portugal, Oct. 2018, pp. 211–215, doi: 10.1109/VLHCC.2018.8506504.
[61] D. Weintrop, "Block-based programming in computer science education," *Commun. ACM*, vol. 62, no. 8, pp. 22–25, Jul. 2019, doi: 10.1145/3341221.
[62] R. S. N. Lindberg, T. H. Laine, and L. Haaranen, "Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games," *Brit. J. Educ. Technol.*, vol. 50, no. 4, pp. 1979–1995, Jul. 2019, doi: 10.1111/bjet.12685.
[63] S. Papadakis and M. Kalogiannakis, "Evaluating the effectiveness of a game-based learning approach in modifying students' behavioural outcomes and competence, in an introductory programming course. A case study in Greece," *IJTCS*, vol. 10, no. 3, p. 235, 2019. [Online]. Available: https://www.inderscienceonline.com/doi/abs/10.1504/IJTCS.2019.102760
[64] A. Mathrani, S. Christian, and A. Ponder-Sutton, "PlayIT: Game based learning approach for teaching programming concepts," *J. Educ. Technol. Soc.*, vol. 19, no. 2, pp. 5–17, 2016.
[65] Y. S. Wong and M. H. M. Yatim, "A propriety multiplatform game-based learning game to learn object-oriented programming," in *Proc. 7th Int. Congr. Adv. Appl. Informat. (IIAI-AAI)*, Yonago, Japan, Jul. 2018, pp. 278–283, doi: 10.1109/IIAI-AAI.2018.00060.
[66] S. S. Oyelere, F. J. Agbo, I. T. Sanusi, A. A. Yunusa, and K. Sunday, "Impact of puzzle-based learning technique for programming education in Nigeria context," in *Proc. IEEE 19th Int. Conf. Adv. Learn. Technol. (ICALT)*, Maceió, Brazil, Jul. 2019, pp. 239–241, doi: 10.1109/ICALT.2019.00072.
[67] T. Orehovački and S. Babić, "Inspecting quality of games designed for learning programming," in *Learning and Collaboration Technologies*, vol. 9192, P. Zaphiris and A. Ioannou, Eds. Cham, Switzerland: Springer, 2015, pp. 620–631, doi: 10.1007/978-3-319-20609-7_58.
[68] F. Duran, C. Arnedo, F. Largo, and R. Carmona, "PLMan: A game-based learning activity for teaching logic thinking and programming," *Int. J. Eng. Educ.*, vol. 33, no. 28, pp. 807–815, 2017.
[69] K. Daungcharone, P. Panjaburee, and K. Thongkoo, "A mobile game-based C programming language learning: Results of university students' achievement and motivations," *Int. J. Mobile Learn. Organisation*, vol. 13, no. 2, p. 171, 2019, doi: 10.1504/IJMLO.2019.098184.
[70] W. Min, B. Mott, K. Park, S. Taylor, B. Akram, E. Wiebe, K. E. Boyer, and J. Lester, "Promoting computer science learning with block-based programming and narrative-centered gameplay," in *Proc. IEEE Conf. Games (CoG)*, Osaka, Japan, Aug. 2020, pp. 654–657, doi: 10.1109/CoG47356.2020.9231881.

[71] C. Kazimoglu, "Enhancing confidence in using computational thinking skills via playing a serious game: A case study to increase motivation in learning computer programming," *IEEE Access*, vol. 8, pp. 221831–221851, 2020, doi: 10.1109/ACCESS.2020.3043278.
[72] C. Malliarakis, M. Satratzemi, and S. Xinogalos, "CMX: The effects of an educational MMORPG on learning and teaching computer programming," *IEEE Trans. Learn. Technol.*, vol. 10, no. 2, pp. 219–235, Apr. 2017, doi: 10.1109/TLT.2016.2556666.
[73] D. Bonner and M. Dorneich, "Developing game-based learning requirements to increase female middle school students interest in computer science," *Proc. Hum. Factors Ergonom. Soc. Annu. Meeting*, vol. 60, no. 1, pp. 380–384, Sep. 2016, doi: 10.1177/1541931213601086.
[74] K. Sharma, J. C. Torrado, J. Gómez, and L. Jaccheri, "Improving girls' perception of computer science as a viable career option through game playing and design: Lessons from a systematic literature review," *Entertainment Comput.*, vol. 36, Jan. 2021, Art. no. 100387, doi: 10.1016/j.entcom.2020.100387.
[75] B. Shneiderman and R. Mayer, "Syntactic/semantic interactions in programmer behavior: A model and experimental results," *Int. J. Comput. Inf. Sci.*, vol. 8, no. 3, pp. 219–238, 1979.
[76] B. Shneiderman, "Teaching programming: A spiral approach to syntax and semantics," *Comput. Educ.*, vol. 1, no. 4, pp. 193–197, 1977.
[77] S. Papert. (1986). *Seymour Papert: On Logo*. [Online]. Available: https://www.youtube.com/watch?v=5uV_lvl4_bE
[78] A. Fay and R. Mayer, "Learning LOGO: A cognitive analysis," in *Teaching and Learning Computer Programming*, 1st ed. London, U.K.: Routledge, 1988, pp. 55–74.
[79] S. M. Glynn, B. K. Britton, M. Semrud-Clikeman, and K. D. Muth, "Analogical reasoning and problem solving in science textbooks," in *Handbook Creativity*, J. A. Glover, R. R. Ronning, C. R. Reynolds, Eds. Boston, MA, USA: Springer, 1989, pp. 383–398, doi: 10.1007/978-1-4757-5356-1_21.
[80] K. Yilmaz, "Constructivism: Its theoretical underpinnings, variations, and implications for classroom instruction," *Educ. Horizons*, vol. 86, no. 3, pp. 161–172, 2008.
[81] J. A. Jiménez-Toledo, C. Collazos, and O. Revelo-Sánchez, "Consideraciones en los procesos de enseñanza-aprendizaje para un primer curso de programación de computadores: Una revisión sistemática de la literatura," *TecnoLógicas*, vol. 22, pp. 83–117, Dec. 2019, doi: 10.22430/22565337.1520.
[82] R. A. Sukamto, H. W. Prabawa, and S. Kurniawati, "Analogy mapping development for learning programming," *J. Phys., Conf.*, vol. 812, Feb. 2017, Art. no. 012109, doi: 10.1088/1742-6596/812/1/012109.
[83] C. Chibaya, "A metaphor-based approach for introducing programming concepts," in *Proc. Int. Multidisciplinary Inf. Technol. Eng. Conf. (IMITEC)*, Vanderbijlpark, South Africa, Nov. 2019, pp. 1–8, doi: 10.1109/IMITEC45504.2019.9015888.
[84] I. Horton, *Ivor Horton's Beginning Java*, 7th ed. Indianapolis, IN, USA: Wiley, 2011.
[85] Y. D. Liang, *Introduction to Java Programming*, 10th ed. Boston, MA, USA: Pearson, 2015.
[86] B. A. Burd, *Java for Dummies*, 5th ed. Indianapolis, IN, USA: Wiley, 2011.
[87] M. Marji, *Learn to Program With Scratch: A Visual Introduction to Programming With Games, Art, Science, and Math*. San Francisco, CA, USA: No Starch Press, 2014.
[88] P. Barry and D. Griffiths, *Head First Programming: A Learner's Guide to Programming Using the Python Language*, 1st ed. Sebastopol, CA, USA: O'Reilly, 2009.
[89] F. Hermans, A. Swidan, E. Aivaloglou, and M. Smit, "Thinking out of the box: Comparing metaphors for variables in programming education," in *Proc. 13th Workshop Primary Secondary Comput. Educ.*, Potsdam, Germany, Oct. 2018, pp. 1–8, doi: 10.1145/3265757.3265765.
[90] J. Radianti, T. A. Majchrzak, J. Fromm, and I. Wohlgenannt, "A systematic review of immersive virtual reality applications for higher education: Design elements, lessons learned, and research agenda," *Comput. Educ.*, vol. 147, Apr. 2020, Art. no. 103778, doi: 10.1016/j.compedu.2019.103778.
[91] K. Busbee and D. Braunschweig, *Programming Fundamentals: A Modular Structured Approach*, 2nd ed. FL, USA: Orange Grove Texts Plus, 2018.
[92] W. Winn, "A conceptual basis for educational applications of virtual reality," Hum. Interface Technol. Lab., Washington Technol. Center, Univ. Washington, Aug. 1993. [Online]. Available: http://www.hitl.washington.edu/projects/education/winn/winn-paper.html~
[93] J. Chapman and P. Rich, "Identifying motivational styles in educational gamification," in *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, 2017, pp. 1–10, doi: 10.24251/HICSS.2017.157.
[94] Y. G. Butler, "Motivational elements of digital instructional games: A study of young L_2 learners' game designs," *Lang. Teach. Res.*, vol. 21, no. 6, pp. 735–750, Nov. 2017, doi: 10.1177/1362168816683560.
[95] I. Cicek, A. Bernik, and I. Tomicic, "Student thoughts on virtual reality in higher education—A survey questionnaire," *Information*, vol. 12, no. 4, p. 151, Apr. 2021, doi: 10.3390/info12040151.
[96] J. Zhao, X. Xu, H. Jiang, and Y. Ding, "The effectiveness of virtual reality-based technology on anatomy teaching: A meta-analysis of randomized controlled studies," *BMC Med. Educ.*, vol. 20, no. 1, p. 127, Dec. 2020, doi: 10.1186/s12909-020-1994-z.
[97] Đ. Đurković and V. Aleksić, "The secondary school Student's interest in virtual reality," in *Proc. 8th Int. Sci. Conf. Technics Inform. Educ.*, 2020, pp. 255–260.
[98] M. Rosenblatt, "A central limit theorem and a strong mixing condition," *Proc. Nat. Acad. Sci. USA*, vol. 42, no. 1, pp. 43–47, 1956, doi: 10.1073/pnas.42.1.43.
[99] S. G. Kwak and J. H. Kim, "Central limit theorem: The cornerstone of modern statistics," *Korean J. Anesthesiol.*, vol. 70, no. 2, p. 144, 2017, doi: 10.4097/kjae.2017.70.2.144.
[100] B. Gribbons and J. Herman, "True and quasi-experimental designs," *Practical Assessment, Res., Eval.*, vol. 5, Dec. 1996, Art. no. 14, doi: 10.7275/FS4Z-NB61.
[101] J. R. Briggs, *Python for Kids: A Playful Introduction to Programming*. San Francisco, CA, USA: No Starch Press, 2013.
[102] L. Hayes, *Beginner Python: The Least You Need to Know*, 1st ed. Scotts Valley, CA, USA: CreateSpace, 2015.
[103] *Student Attitudes toward STEM Survey-Middle and High School Students*, Friday Inst. Educ. Innov., Raleigh, NC, USA, Sep. 2012.
[104] S. G. Hart and L. E. Staveland, "Development of NASA-TLX (task load index): Results of empirical and theoretical research," *Adv. Psychol.*, vol. 52, pp. 139–183, 1988, doi: 10.1016/S0166-4115(08)62386-9.
[105] Y. A. Skaik, "The bread and butter of statistical analysis 't-test': Uses and misuses," *Pakistan J. Med. Sci.*, vol. 31, no. 6, pp. 1558–1559, Dec. 1969, doi: 10.12669/pjms.316.8984.
[106] J. Pallant, *SPSS Survival Manual: A Step by Step Guide to Data Analysis Using IBM SPSS*, 7th ed. London, U.K.: Open Univ. Press, 2020.
[107] E. Norris, N. Shelton, S. Dunsmuir, O. Duke-Williams, and E. Stamatakis, "Virtual field trips as physically active lessons for children: A pilot study," *BMC Public Health*, vol. 15, no. 1, p. 366, Dec. 2015, doi: 10.1186/s12889-015-1706-5.
[108] G. Singh, A. Mantri, O. Sharma, and R. Kaur, "Virtual reality learning environment for enhancing electronics engineering laboratory experience," *Comput. Appl. Eng. Educ.*, vol. 29, no. 1, pp. 229–243, Jan. 2021, doi: 10.1002/cae.22333.
[109] L. L. Garber, E. M. Hyatt, and Ü. Ö. Boya, "Gender differences in learning preferences among participants of serious business games," *Int. J. Manage. Educ.*, vol. 15, no. 2, pp. 11–29, Jul. 2017, doi: 10.1016/j.ijme.2017.02.001.
[110] M.-T. Wang and J. L. Degol, "Gender gap in science, technology, engineering, and mathematics (STEM): Current knowledge, implications for practice, policy, and future directions," *Educ. Psychol. Rev.*, vol. 29, no. 1, pp. 119–140, Mar. 2017, doi: 10.1007/s10648-015-9355-x.
[111] V. Riemer and C. Schrader, "Learning with quizzes, simulations, and adventures: Students' attitudes, perceptions and intentions to learn with different types of serious games," *Comput. Educ.*, vol. 88, pp. 160–168, Oct. 2015, doi: 10.1016/j.compedu.2015.05.003.
[112] C. Erolin, L. Reid, and S. McDougall, "Using virtual reality to complement and enhance anatomy education," *J. Vis. Commun. Med.*, vol. 42, no. 3, pp. 93–101, Jul. 2019, doi: 10.1080/17453054.2019.1597626.
[113] V. Zirawaga, A. Olusanya, and T. Maduku, "Gaming in education: Using games as a support tool to teach history," *J. Educ. Pract.*, vol. 8, no. 15, pp. 55–64, 2017.
[114] C.-Y. Hung, J. C.-Y. Sun, and J.-Y. Liu, "Effects of flipped classrooms integrated with MOOCs and game-based learning on the learning motivation and outcomes of students from different backgrounds," *Interact. Learn. Environ.*, vol. 27, no. 8, pp. 1028–1046, Nov. 2019, doi: 10.1080/10494820.2018.1481103.
[115] M. Berkman, G. Çatak, and M. Ç. Eremektar, "Comparison of VR and desktop game user experience in a puzzle game: 'Keep talking and nobody explode,'" *AJIT-e, Online Acad. J. Inf. Technol.*, vol. 11, no. 42, pp. 180–204, Oct. 2020, doi: 10.5824/ajite.2020.03.008.x.

[116] A. K. Rao, S. Chandra, and V. Dutt, ‘‘Desktop and virtual-reality training under varying degrees of task difficulty in a complex search-and-shoot scenario,’’ in *Proc. Int. Late Breaking Papers, Virtual Augmented Reality*, vol. 12428, C. Stephanidis, J. Y. C. Chen, and G. Fragomeni, Eds. Cham, Switzerland: Springer, 2020, pp. 421–439, doi: 10.1007/978-3-030-59990-4_31.

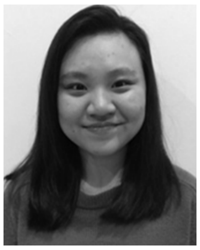
[117] A. K. Rao, B. S. Pramod, S. Chandra, and V. Dutt, ‘‘Influence of indirect vision and virtual reality training under varying manned/unmanned interfaces in a complex search-and-shoot simulation,’’ in *Advances in Human Factors in Simulation and Modeling*, vol. 780, D. N. Cassenti, Ed. Cham, Switzerland: Springer, 2019, pp. 225–235, doi: 10.1007/978-3-319-94223-0_21.

[118] H. Rivera-Flor, K. A. Hernandez-Ossa, B. Longo, and T. Bastos, ‘‘Evaluation of task workload and intrinsic motivation in a virtual reality simulator of electric-powered wheelchairs,’’ *Proc. Comput. Sci.*, vol. 160, pp. 641–646, Jan. 2019, doi: 10.1016/j.procs.2019.11.034.

[119] K. Zengel, ‘‘The electromagnetic analogy of a ball on a rotating conical turntable,’’ *Amer. J. Phys.*, vol. 85, no. 12, pp. 901–907, Dec. 2017, doi: 10.1119/1.5002686.

[120] (2020). *Calculus Analogies*. [Online]. Available: https://nrich.maths.org/7087

[121] C. Botella, J. Osma, A. G. Palacios, V. Guillén, and R. Baños, ‘‘Treatment of complicated grief using virtual reality: A case report,’’ *Death Stud.*, vol. 32, no. 7, pp. 674–692, Aug. 2008, doi: 10.1080/07481180802231319.



CHYANNA WEE received the degree (Hons.) in computer networking and security from Sunway University, where she is currently pursuing the master’s degree in computer science. She is currently with the Human-Machine Collaboration Research Centre (HUMAC). Her research interests include human–computer interaction, with a focus on perceptual interfaces (virtual reality and haptics), human-centered artificial interfaces, and usable programming.

KIAN MENG YAP (Senior Member, IEEE) is currently the Founder of House of Multimodal Evolution (H.O.M.E.) Laboratory, which is to provide novel and innovative multimodal applications and communications toward the technology world. He is also the Founder of the Human-Machine Collaboration Research Centre (HUMAC) that aims to be nation’s main technology hub and to demonstrate its commitment to sustainable development. He has spent 20 years working in data networks (IT), telecommunications, computer networking, haptics, manufacturing industries, and electronic control system in machinery. His research project on DHVE is meant to have multi-sensory feedback data, such as voice, audio, and force over the fixed and wireless network. He is a principal investigator for a few projects that are supported by the Ministry of Higher Education (MOHE), ERGS, MCMC, Lancaster University, U.K., PPRN, industrial partners, and Sunway University Internal Grant, respectively. His current research interests include haptic over the distributed virtual environment (DVE) in HCI environment, haptics and odor sensing, drones, odor sensing/tracking, tele-haptics, tele-robotics, materials sensing, assistive technology for limited vision and hearing impaired, and AR&VR.

WOAN NING LIM (Senior Member, IEEE) received the bachelor’s degree (Hons.) in computer science and the Master of Engineering degree in electrical engineering from the Universiti Teknologi Malaysia, in 1998 and 2000, respectively. She is currently a Senior Lecturer with the School of Engineering and Technology, Sunway University. She is also a Researcher with the Human-Machine Collaboration Research Centre (HUMAC). She has vast experience in the IT industry, worked in several multinational organizations, such as Shell IT International, DHL Asia Pacific IT Services, and Standard Chartered Scope International. Her research interests in human–computer interaction focusses on virtual reality and augmented reality, mobile computing, machine learning, and image processing. She is involved in a few research grants projects such as the Sunway Internal Grant and the MOHE Fundamental Research Grant Scheme (FRGS).