Paper ID #37431

Learning Robot Programming Anywhere: VEXcode VR (Other)

Arif Sirinterlikci

Arif Sirinterlikci is a university professor of industrial and manufacturing engineering at Robert Morris University. His teaching and research interests lie in manufacturing engineering, specifically in industrial automation and robotics, CAD/CAE/CAM, 3D scanning and printing, medical manufacturing, and entertainment technology. In addition, he has recently developed courses in different areas of Industry 4.0 including Industrial Internet of Things (IIoT) and Mixed Reality (MR) for Industry.

Jason McKenna

Yuhan Lin

Yuhan Lin is a doctoral student in the Technology, Learning and Leadership PhD program at the University of Maryland's College of Education. He got his undergraduate degree from the University of Georgia, double majoring in Mathematics and Mathematics Education. He also has a master's from University of Pennsylvania Graduate School of Education in Learning Science and Technology. As a lifelong programmer, he holds a firm belief in the importance of computer programming education. His recent work involves creating a taxonomy for block-based programming environments. His research focuses on exploring ways to bridge the gap between block-based and text-based programming and using physical computing as a means to support computer science programming education.

Raina Oravec

Raina Oravec is an Educational Analyst at VEX Robotics. She graduated from Chatham University with a Bachelor of Arts in Policy Studies with a focus in after school education development. Throughout college Raina worked with a small private school to bring more hands-on approaches to STEM education for K-8 students. Previously, Raina worked in STEM education, user data analytics and as a Customer Relations Management developer for high education.

Lauren Harter

Lauren Harter is a Senior Educational Developer at VEX Robotics and has a wide range of experience in education. From teaching in the high school setting to developing materials that teachers use in numerous countries, Lauren's experiences have shaped her contributions to the educational community. Lauren received a double bachelors in Mathematics and Secondary Mathematics Education from Duquesne University in 2016. Shortly after, she began teaching high school mathematics at Serra Catholic High. For two years, she taught 9-12th grade Algebra I, Algebra II, Trigonometry, and Calculus to a wide range of students. Lauren is nearing the end of her Doctoral studies and is conducting research in teaching practices that promote conceptual understanding in mathematics and teacher quality.

Learning Robot Programming Anywhere: VEXcode VR

Introduction

In January of 2016, then President Barack Obama addressed the changing demands of the 21st century economy, and identified computer science (CS) as the "new basic skill required of contemporary students"[1]. To meet the challenge of his time, he announced a plan to give all students access to CS education. President Obama's identification of the importance of CS education mirrors the attitudes and opinions of many educators, and also that of the nation's parents and students themselves. This is emphasized by the findings of a recent study conducted by Google; 82% of students were at least somewhat interested in learning CS, with 84% of parents citing CS as being at least as important as required (and more familiar) subjects like math and reading[2].

CS education is a part of STEM (science, technology, engineering, math) education. The National Science and Technology Council's Committee on STEM Education put forth a report in 2018 to outline a federal strategy for STEM education. This report notes that, "The character of STEM education itself has been evolving from a set of overlapping disciplines into a more integrated and interdisciplinary approach to learning and skill development. This new approach includes the teaching of academic concepts through real-world applications and combines formal and informal learning in schools, the community, and the workplace. It seeks to impart skills such as critical thinking and problem solving along with soft skills such as cooperation and adaptability."[3] This national focus on STEM learning has been accompanied with increased research and innovation in educational settings on how to better incorporate technology into the classroom for STEM topics.

Robotics provides a hands-on way for students to explore STEM concepts, resulting in an increase in its use in recent years. Studies have shown that Educational Robotics can be an effective tool to teach CS while also helping to broaden participation goals[4,5]. Recent advances in Educational Robotics have lowered costs and increased ease of use, making them more accessible to students and progressively turned to as a reliable way to learn CS concepts.

As such, the connection between CS and robotics is clear; students have the ability to program their robots to perform complex tasks, both in the classroom and on competition fields. The ability to effectively teach generalizable CS skills, while simultaneously offering ways to help diversify the students that enter these fields, makes Educational Robotics a significant contributor to the integration of computational thinking into schools and the 'Computer Science for All' movement.

Unfortunately, the COVID-19 pandemic caused widespread global disruption to in-person learning, affecting nearly all students worldwide[6]. Hands-on learning experiences were suspended, which was a foundational portion of most robotic STEM curriculum, including the curriculum used by the VEX educational robotics line. Remote learning solutions were needed to quickly provide a virtual learning environment that could still help students engage with STEM topics in an authentic, meaningful way. Virtual learning of physical computing elements also has the feature of making the concepts, curricula and their supporting environments accessible to students who may not otherwise have access to physical computing or robotics platforms. While there are some low-cost physical computing devices such as micro:bit and lilypad arduino (less than 50 US dollars per device), many physical computing kits, such as LEGO and VEX robotics, have costs that run into the hundreds. A result of shifting physical robotics to online is that it becomes more possible for each learner to have their own environment to learn with, rather than needing to share time with classmates. Although some computer science classrooms decided to ship all of their physical computing components to each individual student during the pandemic, a one-to-one physical computing device ratio would not be feasible for more expensive equipment. In addition, having a virtual programming environment gives each student the ability to have a virtual device. Another unique feature of virtual robotics is that it could allow students to make changes of their designs in different environments with just a click. It also eliminates the physical error a student may have such as loose screws or frictions from the materials. In a non-dynamic virtual playground, a virtual robot should perform exactly the same in any run with the same program. Thus, it would help students to reduce one area of error when debugging.

VEX Robotics quickly created VEXcode VR (hereafter simply referred to as “VR”) to respond to the global disruption caused by the COVID-19 pandemic with a virtual robot platform that could be used in similar ways as a physical robot. Even though, there are other virtual robotics platforms like Virtual LEGO EV3 or Blockly for Dash and Dot Robots, they do not correlate to VEX Robots, a popular Educational Robotics system worldwide.

This paper will review the usage data collected by the VR platform to gain insights into how this virtual substitute was during this global disruption. Curriculum based on multiple case studies will also be presented which provide context for how teachers implemented VR in their remote learning environments. The two primary research questions for this paper are as follows:

1. What insights can usage data and teacher case studies reveal about student learning with VR following the COVID-19 outbreak?
2. What insights can students provide with their use of VEXcode VR?

Prior Work

Learning computer science, similar to subjects like biology and chemistry, helps to teach students about the world around them while also teaching them foundational skills that can be applied to a wide range of careers and opportunities for innovation. However, learning to program can be difficult for many students, with some estimating that 10 years is needed to turn a novice into an experienced programmer[7].

Difficulty in learning programming is not confined to a particular age – elementary, middle, and high school novice programmers all show problems with learning programming[8]. Nearly all of the research on debugging has found that students have a difficult time predicting the output of a program, debugging a program, and writing original programs[8]. We will explore some of the difficulties that students may have when learning programming. Using the example earlier of biology and chemistry, students in those classes have been exposed to biology and chemistry all of their lives. Yes, many students are exposed to computing devices like phones and iPads, but these devices don’t teach one computer science. Also, students often carry a fundamental misunderstanding with them into a programming class – the assumption that the computers "understand" programming languages in the same way that we humans understand spoken languages[8]. Humans are able to understand inference and nuance in spoken language, whereas computers can only do exactly what they are told.

Students can also have difficulties learning programming due to the inexperience of their teachers. Many of the educators teaching CS right now have not learned or taught CS before[9]. This is problematic because a well-prepared and knowledgeable teacher is the most important school-side factor in student learning[10].

Block-based coding languages have been developed to help novice programmers. One of the primary motivations behind the creation of block-based coding languages is that many young students find programming within text-based environments too difficult, as these students find both the typing and the syntax difficult to master[11]. As a result, block-based programming languages have been designed for students as young as preschoolers, but most are aimed for students between the ages of 8 and 16. Many of the block-based programming environments also start to support the transition to text-based programming which can help students to advance their programming abilities. Block-based programming presents several advantages for beginning programmers[11]:

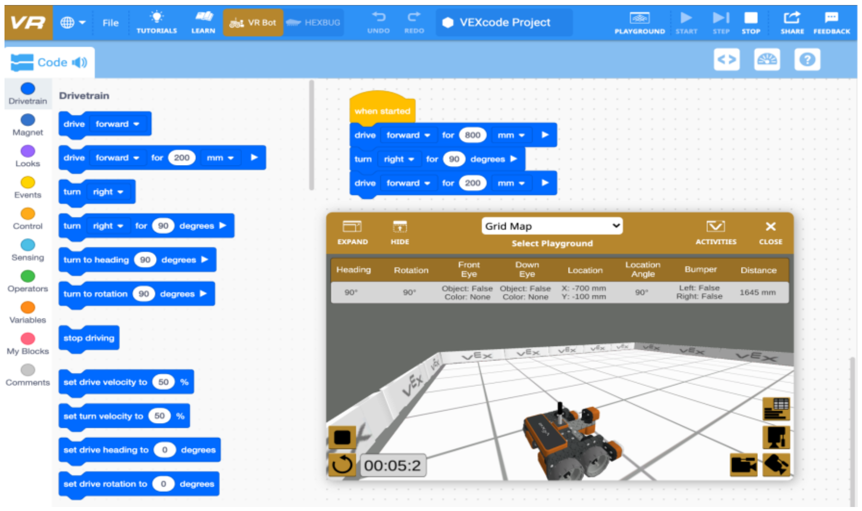
1. Readability: Block-based programming languages have commands that are much easier to read than text-based commands.
2. Memorization: Block-based programming languages have all of their commands visible to the user. With text-based programming languages, commands often need to be memorized. Additionally, users have to memorize the syntax that is associated with the text commands. There is no syntax associated with block-based commands.
3. Typing/Spelling: Younger students struggle with using a keyboard. Spelling mistakes become compiler errors with a text-based programming language. Block based programming languages use drag and drop. Therefore, no typing or mastery of spelling is needed.

Teaching Programming with Educational Robots

Educational robotics are an increasingly popular way to introduce and teach STEM and CS to students[3]. Educational robotics can also serve as an engaging way to connect CS to the "T" and "E" in STEM[12]. Educational robotics can be an effective tool to teach CS while also helping to broaden participation goals in CS[4,5]. Recent advances in educational robotics have lowered costs and increased ease of use, making them more accessible to students and progressively become a reliable way to learn CS concepts. Students as young as 4 years old have been shown to successfully build and program simple robotics projects[12]. Educational robotics can help with the learning of CS in making abstract concepts easier to understand by allowing students to see the relationship between their code and the actions of the robot.

The connection between CS and robotics is clear; students have the ability to program their robots to perform complex tasks, both in the classroom and on competition fields. While the performance of complex tasks may be the end, the means involve decomposing these tasks into smaller parts and then iteratively building them together to create a solution.

These connections are seen in both physical and virtual robot environments. A virtual robotic environment is an environment where the robot functions as a real robot would, including sensor feedback and drivetrain commands, but in a sanitized environment. This sanitized environment removes instances such as environmental factors that can change the outcome of sensor feedback or robotic driving, in addition to the robot not having to be physically reset after each run (Figure 1).



```
Drivetrain
drive forward
drive forward for 200 mm
turn right
turn right for 90 degrees
turn to heading 90 degrees
turn to rotation 90 degrees
stop driving
set drive velocity to 50 %
set turn velocity to 50 %
set drive heading to 0 degrees
set drive rotation to 0 degrees
```

```
when started
drive forward for 800 mm
turn right for 90 degrees
drive forward for 200 mm
```

Heading	Rotation	Front Eye	Down Eye	Location	Location Angle	Bumper	Distance
90°	90°	Object: False Color: None	Object: False Color: None	X: -700 mm Y: -100 mm	90°	Left: False Right: False	1645 mm

Figure 1: The VR Robot

Virtual environments can provide several unique learning opportunities such as[12]:

- Quick review of their robot performing an action
- Pausing the learning environment and quickly remedying an error
- Having access to virtual environments outside of their classroom
- Less time devoted to correcting mechanical errors and more time spent on programming

CS Curriculum

CS curriculum, similar to other STEM subjects, often takes the form of instructional text to explain concepts, supported by worked examples to provide examples and scaffolding for students[13]. Recently, subgoal labels have been utilized to help foster generalizable CS knowledge[14]. Subgoal labeling is the process of giving a name to a group of steps, in a step-by-step description of a process, to explain how the group of steps achieve a related subgoal.

Because of the abstract nature of CS, students benefit from a CS curriculum that contextualizes learning objectives through the creation of a shared goal. Ensuring that students and teachers share this similar focus is an important part of teaching and learning. Furthermore, this approach centers the lesson around student construction of a solution to the problem[15].

Once the shared goal is established, direct instruction can then help students to get started by providing step-by-step directions on how to approach the lessons within each unit[16]. This careful sequencing of learning emphasizes understanding, instead of superficially covering multiple topics.

Research then tells us that once the previously applied direct instruction foregrounds the CS skills and concepts needed to solve the shared goal, the scaffolding should be removed from the students – thus requiring students to transfer their knowledge to a novel situation[17].

Methods Used in This Study

This paper employed a simple quantitative survey given to a small group of students participating in a VEX VR pilot study. In addition, survey participants' qualitative feedback as well as analytics data from more than 326K users on the Internet were also presented in this section. A second survey was also given to another small group of workshop participants resulting in similar feedback.

The data presented in this section is provided by Google Analytics *to the authors*. As VEXcode VR is entirely browser-based, there are a number of different metrics which provide insight into how this virtual robot environment has been used globally. Since its launch in April 2020, there has been an increase in VR users monthly, which have combined to over 2.6 million users in more than 150 countries.

A project is a program that students create for a lesson or challenge. Projects do not have to be saved in order to run, but a saved project is downloaded for a user to come back to at a later time. There were over 4 million saved projects. Because VR is entirely browser based, editing a project and testing it happens immediately by selecting "START" (Figure 2).

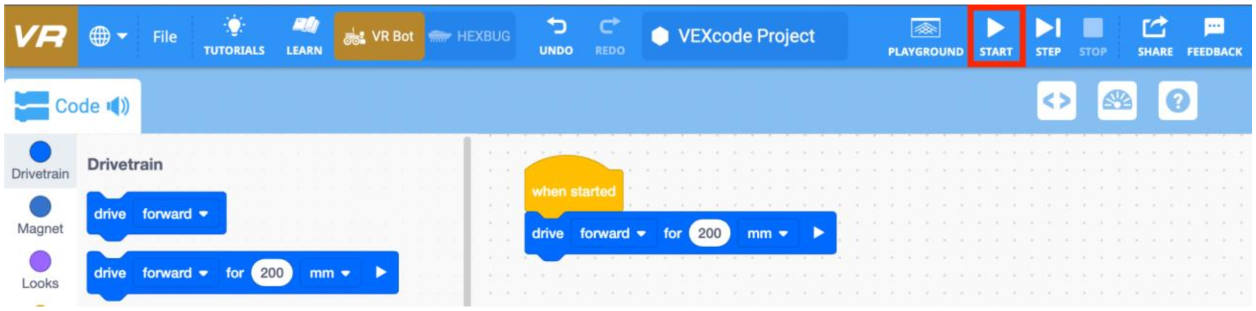


Figure 2: VEXcode VR 'START' button in the Toolbar

There have been more than 150 million project runs in the interface software, indicating that students tested their projects at frequent intervals. Due to this immediate feedback loop, students had the opportunity to experiment and iterate at a much faster pace compared to working with a physical robot. This iterative process is a good indication for student learning as multiple iterations have been shown to maintain student engagement and interest[18].

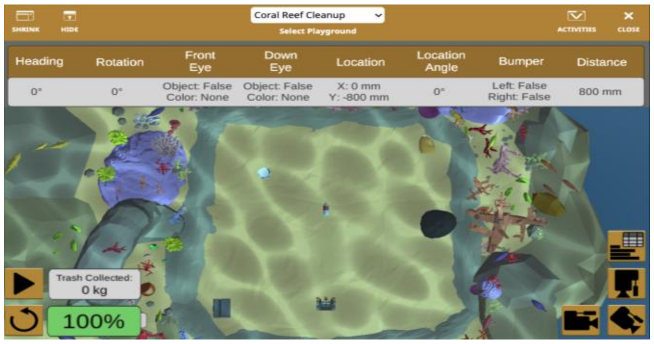
VEXcode VR Data (as of 2/7/2022)	
Lifetime Project Runs	150,080,037
Lifetime Coding Sessions	10,823,506
Lifetime Total User Coding Time	More than 3.8 million hours
Lifetime Number of Users	2,659,928
Number of Countries Used	156

Table 1: VEX VR Usage Data

Since all of the curriculum is web-based, Google Analytics can also be used to capture data. To date, there have been 326K unique users of the VEXcode VR curriculum. VEXcode VR also has supplemental activities, which have acquired just over 1 million users.

Pilot Study Protocol

In addition to the summer workshops offered at the academic collaborator's institution, the authors conducted one-hour interview sessions with 8 participants between the ages of 13 and 15. During the interview, participants were introduced to the VEX VR platform which allows the user to write programs for a virtual robot to carry out (Figure 1 and 3). During the first 10 minutes of the interview, participants were instructed to play around with the environment to see what it could do. Participants were then shown a short video introducing the "coral reef cleanup" challenge they would spend the next 20 minutes working on. For this challenge, participants need to program their robot to collect garbage scattered around the seafloor (Figure 3). After 20 minutes of the task, each participant was asked to complete a survey and then we conducted a brief interview to learn about their experience of programming their virtual robot.



Heading	Rotation	Front Eye	Down Eye	Location	Location Angle	Bumper	Distance
0°	0°	Object: False Color: None	Object: False Color: None	X: 0 mm Y: -800 mm	0°	Left: False Right: False	800 mm

Figure 3: VEXcode VR Playground - (Coral Reef Cleanup Playground)

Each of *the pilot study* sessions were video and screen recorded, and then transcribed. The transcript and videos were systematically analyzed, specifically attending to ways learners navigated the virtual nature of the robot and their experiences programming a virtual version of a physical robot. For each participant, the authors also tracked which categories of blocks they used during the study.

Pilot Study Demographic Data

The participant group contains 4 males and 4 females. All participants had previous Scratch programming experience. Seven out of eight participants also had code.org experience, while five of them also had Python experience. Six out of eight participants also had experience with at least one of the physical computing components such as micro:bit, LEGO or Sphero. Participants were recruited through a community listserv, with all participants self-identifying as Asian and not Hispanic or Latino.

Findings of the Pilot Study on VEX VR

The survey results showed us that students generally felt the virtual robotics platform was fun and easy to use and it would help them to program a robot in the real-world. On a scale of one (strongly disagree) to seven (strongly agree), here is their result.

	Statement	Average	STDev
1	I think VEX VR was easy to use	5.625	0.517
2	VEX VR was easier than Scratch	4.75	0.886
3	Creating the programs was frustrating	3.625	1.302
4	Programming the VEX VR was fun	5.625	1.187
5	VEX VR is more fun than Scratch	4	1.069
6	VEX VR will help me program a robot in the real-world	5.625	0.916
7	I feel like I was able to get the robot to do what I wanted it to do	4.75	1.164

Table 2: Survey Result from Students

The virtual robotics environment used here is easy and fun to use and could help program a real robot. However, when compared to Scratch, it may not be much more fun and creating the programs could still be frustrating for some students.

Many participants mentioned the benefit of block-based programming environment (BBPE) in VEX VR. P1 said it is easy when compared to Java, as Java is complex and not similar to the natural language. P3 said it is a clean interface and it is easy to understand intuitively. “Blocks are easier to understand”. P4 said that he was able to control the robot to do what he wanted to and blocks look like Scratch, thus looks nice. He also thought the playground with animation looked nice. P7 said that it is very similar to code.org and the animation is cool and he said “Programming VEX is easy, blocks are there, just need to put in numbers.” He also gave a big compliment that it is a good way to get people into both programming and robots. P8 said Scratch and VEX VR are similar. P1 also mentioned that she would want to see more functions that are different from Scratch. Thus, using a BBPE for the virtual robotics would be helpful to students. Besides the feedback on programming interface and its content, qualitative user feedback pointed out the benefits of using a virtual robot and engaging students in robotics with help from teachers prior to using virtual platforms.

Benefit of Using a Virtual Robot

Many participants mentioned the benefits of using a virtual robot including multi-view environment, money-saving factor, and the similarity of programming a real robot.

P1 said that “you can see a robot in a virtual place and a lot of perspectives”. Both P6 and P7 also mentioned the multi-view camera and he was able to drag and see from different angles. Thus, it is fun for him to use. P1 also mentioned that she was able to save money when using virtual robotics instead of a physical robot. P2 mentioned that programming a virtual robot is really similar to programming a real robot. It would be beneficial to program real robots. P5 said it is nice to see the end product and be able to see the drive. P8 said that the physical robots would be more fun and she used physical robots before. She would like to have more hands-on experience with a real robot and VEX VR would help robots.

When comparing a virtual robot to some other physical computing component such as microbit, P1 also mentioned that micro:bit was not as versatile as robot. Robots are much more fun. P3 also said VEX virtual robots are more interesting to program than a LED robot which the researcher believed refers to micro:bit.

VEXcode VR is oftentimes used as a supplement when students are learning with physical robots, but VEXcode VR is also very effective as its own entity. The fact that students can focus on learning coding concepts such as: project flow, sequencing, and many other concepts without also having to learn physical aspects of robotics is beneficial. Other physical aspects of robotics that sometimes students have to learn in tandem to coding concepts include how to adapt sensors or other physical robotic hardware depending on environment changes. Using VEXcode VR enables students to focus on, and truly learn, certain concepts at one time.

Robotics Related Term Need Teacher in Place

Although the authors saw a lot of benefit from VEX VR, some functionality would probably need some more explanation such as sensing. From a researcher's standpoint, the authors can see the issue of sensing, especially for students who had never used a robot. P3 mentioned “Harder things would be sensing, operator, variable, others are easy.” P6 believed that VEX VR could help to program a real robot and it would give the user more knowledge on what robots do. Even though sensing is hard, he knows what sensing is supposed to do, but does not know how to use it. P7 mentioned that she didn’t know how to use a bumper. P8 said she didn’t know the controls and needed guidance. Thus, it would be better to have teachers as guidance on the side as all these unfamiliar blocks are mainly robotics related.

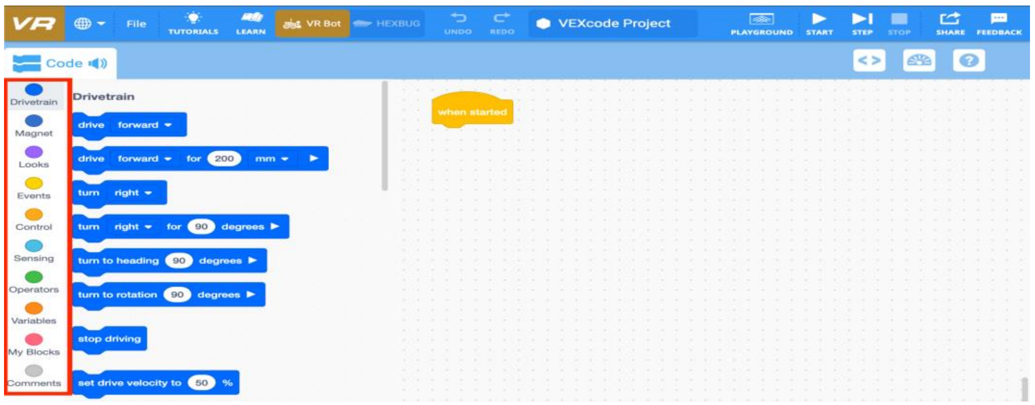
P4 said “It [coral reef task] is easy but challenging.” P5 commented that it was not easy, but somewhat easy since he had prior experience with BBPE. P5 used brute force method when programming instead of any loops. Many students mentioned that they would like to have a teacher in place to guide them and answer additional questions they may have.

Discussion

VEXcode VR is for experimenting with different ways to program a robot. To encourage self-directed learning, new projects can be created and played in seconds, immediate feedback is provided, and both sensor data and the program execution is made visible for the learner. VEXcode VR is a way to enrich the CS experience for students after they have discovered the excitement of educational robotics with other physical platforms such as VEX. Ease of use, immediate feedback, and making learning visible are the three big points seen by the users of the VEX VR platform.

Ease of Use

VEXcode VR is web-based, so launching VEXcode VR is easy. The user interface makes navigation simple. Commands are divided into categories avoiding a potentially overwhelming list of commands (Figure 4). Commands are also color-coded so users can easily find related blocks. The programming area is always visible; inviting learners to begin coding. VEXcode VR utilizes prebuilt robots and drivetrain commands. This allows users to code a robot to move within moments.



```
drive forward ▾
drive forward ▾ for 200 mm ▾ ▶
turn right ▾
turn right ▾ for 90 degrees ▶
turn to heading 90 degrees ▶
turn to rotation 90 degrees ▶
stop driving
set drive velocity to 50 %
when started
```

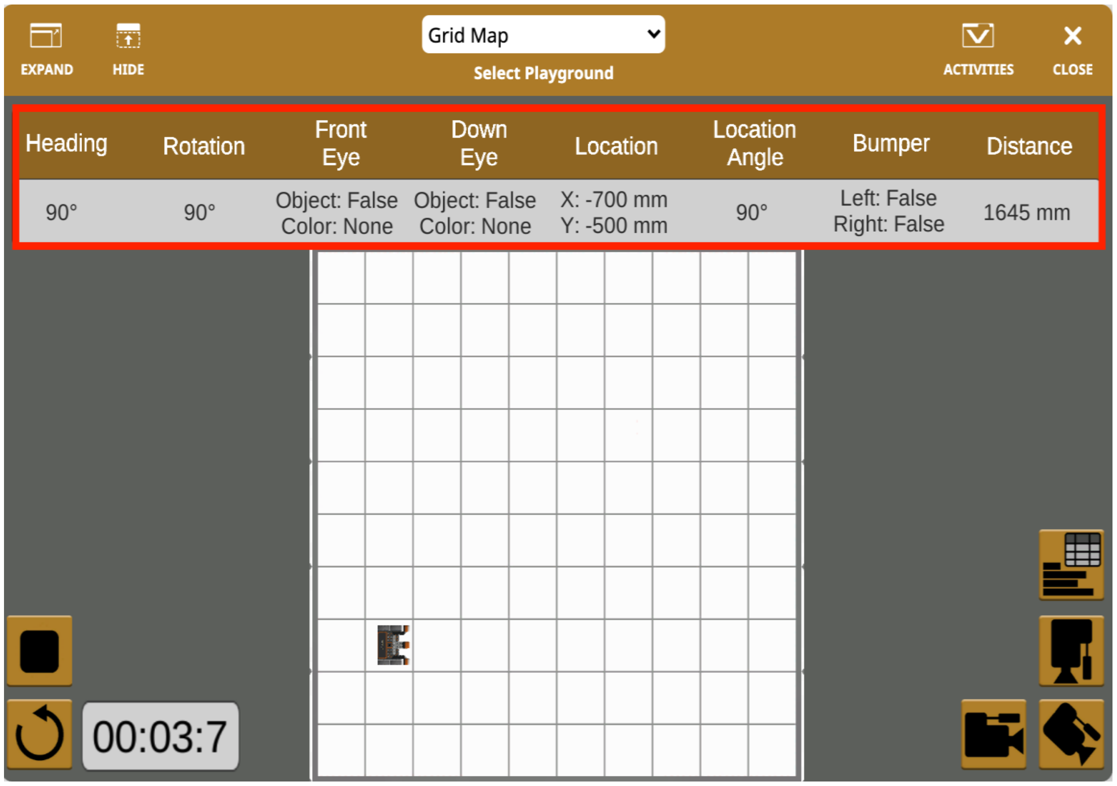
Figure 4: VEXcode VR Categorized Commands

Immediate Feedback

VEXcode VR encourages experimentation and play. When students run a project, they can immediately see if their robot produced the intended result. It is easy for teachers to provide feedback. A project in VEXcode VR will always run the same way–this doesn't always occur with a physical robot. This allows teachers and students to focus on the logic of the programming, not the physics of the robot or the field where the robot is being run. The VR Robot always begins in the same location and isn't impacted by friction if its batteries are running low or fully charged. Learners can add blocks while their project is running, stop the project at any point, and reset their virtual playground with one click. Blocks that are not connected to the main stack are just ignored when the project is run. There are no syntax errors in VEXcode VR. Students may make logical mistakes when coding, but they will not become frustrated that their projects will not compile and run, as they might when coding a physical robot. VEXcode VR's ability to provide immediate feedback and its ease of use encourage students to learn while they code, and supports a bottom-up approach to writing projects where small snippets of code are created, tested and then combined into larger behaviors.

Learning Made Visible

The Playground window in VEXcode VR contains a dashboard that displays all of the sensor data from the VR Robot. Anytime the VR Robot runs, learners can see the sensor data update in real-time, providing them with information on how the data can be used (Figure 5).

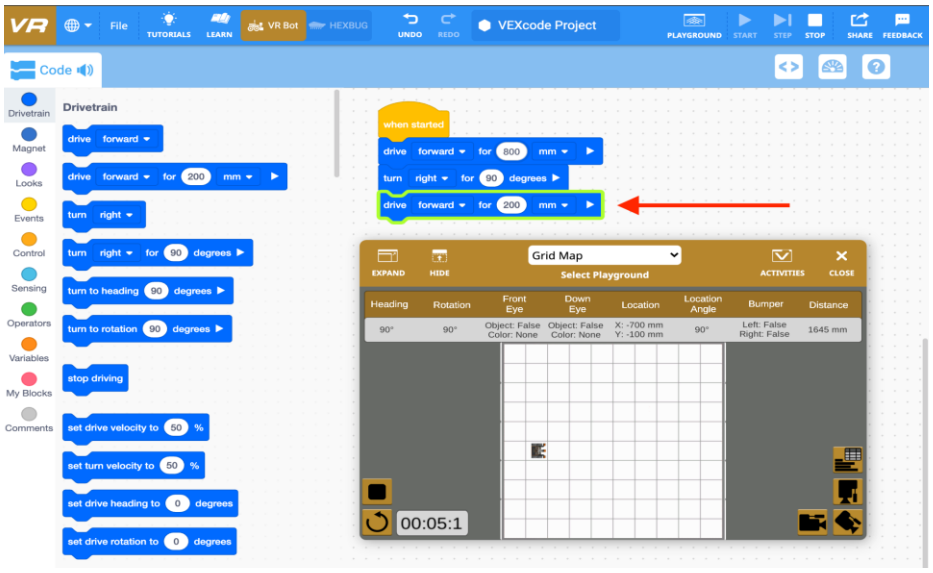


Heading	Rotation	Front Eye	Down Eye	Location	Location Angle	Bumper	Distance
90°	90°	Object: False Color: None	Object: False Color: None	X: -700 mm Y: -500 mm	90°	Left: False Right: False	1645 mm

Figure 5: VEXcode VR Sensor Dashboard

This real-time data can help learners make abstract concepts (e.g. How does my robot make a decision?) more visible and concrete.

VEXcode VR also highlights the blocks within a project when those blocks are being executed. This feature allows learners to observe the program flow of their projects. When a VEXcode VR project is running, the block that is being executed is surrounded by a glowing green border. This feedback helps learners to understand why the VR Robot is performing a particular behavior (Figure 6).



```
when started
drive forward for 800 mm
turn right for 90 degrees
drive forward for 200 mm
```

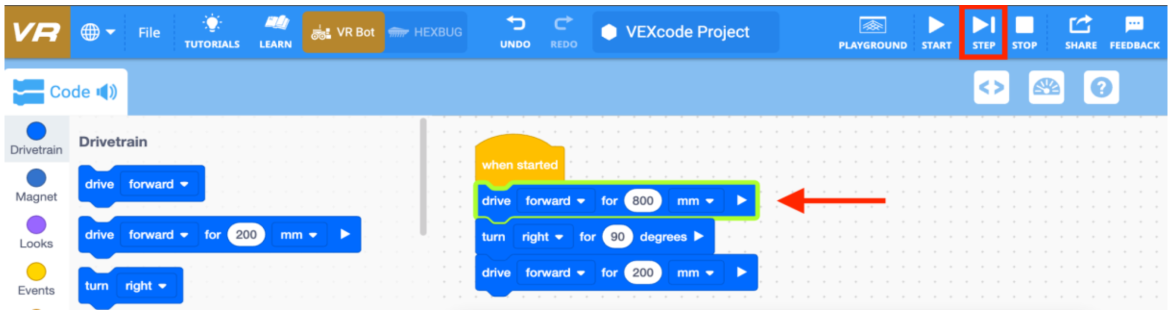
Drivetrain

```
drive forward
drive forward for 200 mm
turn right
turn right for 90 degrees
turn to heading 90 degrees
turn to rotation 90 degrees
stop driving
set drive velocity to 50 %
set turn velocity to 50 %
set drive heading to 0 degrees
set drive rotation to 0 degrees
```

Heading	Rotation	Front Eye	Down Eye	Location	Location Angle	Bumper	Distance
90°	90°	Object: False Color: None	Object: False Color: None	X: -700 mm Y: -100 mm	90°	Left: False Right: False	1645 mm

Figure 6: VEXcode VR Highlighted Block

Students can also Step through their projects by using the step button in the toolbar. When a user steps through a project, the block is first highlighted in green to show that it is the next command to be run. When the user selects Step again, the action of the block gets executed and the green border blinks (Figure 7).



```
when started
drive forward for 800 mm
turn right for 90 degrees
drive forward for 200 mm
```

Drivetrain

```
drive forward
drive forward for 200 mm
turn right
```

Figure 7: VEXcode VR ‘STEP’ feature

The step feature provides students with a visual representation of the flow of the commands as they are being executed within a project. This provides students with important visual cues when they are trying to troubleshoot or design a program.

VEXcode VR projects cannot be downloaded to a physical, VEX robot at this stage. This can lead to another learning opportunity for teachers and students as they need to make slight adjustments to the code for their virtual robot to work with a physical robot. This opportunity also enables students to generalize their understanding and apply it to different contexts, which is a desirable goal for more advanced students.

VEXcode VR Curriculum

VEX's Computer Science with VEXcode VR is a curriculum that anyone can teach and everyone can learn. Designed for students of different interests and experience levels, students learn core CS concepts as they code the VR Robot. This curriculum is available at cs.vex.com. Curriculum is designed to allow students to work independently through each lesson - allowing it to be implemented in multiple ways (e.g. blended, synchronous, asynchronous).

Lessons are structured in the following manner:

Step 1 – Create a shared goal. For students to solve open-ended tasks, both the teachers and the students must be like-minded on how the task is to be solved. Beginning a lesson by putting the students and teacher "on the same page" limits student frustration and centers the lesson around student construction. This shared goal is established with an introductory video at the beginning of each unit. This video sets up topic areas, tasks, and task contexts that puts teachers and students "on the same page." Research tells us that ensuring that students and teachers share this similar focus is an important part of teaching and learning[17]. Furthermore, this approach centers the lesson around student construction of a solution to the problem[15].

Step 2 – Activate Prior Knowledge.Direct instruction at the outset of an open-ended task foregrounds the skills and concepts needed to solve the task[16]. When direct instruction is sequenced correctly, the emphasis is on student understanding, not the covering of facts or guessing and checking.

Step 3 – Check for understanding. Formative assessment embedded throughout the lesson allows teachers to understand and identify student needs, allows teachers to differentiate their teaching, and create student self-efficacy. Formative assessment provides teachers with the means and the opportunity to acquire the right type of information needed to guide their instruction, thus leading to higher learning outcomes[19]. Additionally, teachers using formative assessment are better prepared to meet diverse students' needs – thus helping to achieve a greater equity of student outcomes[20].

Step 4 – Allow students to apply what they learned on their own. Scaffolding is taken away at the end of the lesson, providing students an opportunity to apply what they have learned to a different context[17].

Conclusion

As President Barack Obama mentioned the importance of CS education, as well as research that suggests the incorporation of technology in the classroom for STEM topics, Educational Robotics emerges as a contributor for incorporating CS into educational settings. This paper reviewed the usage data collected by the VR platform and interviews with students to gain insights into how this virtual substitute was during this global disruption. The two primary research questions were:

1. What insights can usage data and teacher case studies reveal about student learning with VR following the COVID-19 outbreak?
2. What insights can students provide with their use of VEXcode VR?

VEXcode VR has been shown to encourage self-directed learning, provide immediate feedback, provide easy access to students and schools, and show a strong connection between coding a physical and virtual robot.

From the student interviews, themes of flexibility, continuity with physical robots, and increased CS learning were identified as important to teaching with technology in such uncertain and challenging circumstances.

Not only is VEXcode VR an effective tool as its own entity, there is also teacher and student materials, such as curriculum, as well as an educator certification that can be used in conjunction with the software to teach CS concepts. Using VEXcode VR as a learning tool by itself, or in combination with physical robots, could provide a well-rounded and flexible learning environment where students can learn CS concepts and STEM topics in an authentic, meaningful way.

References

[1] M. Smith. *Computer Science For All | whitehouse.gov*. Obama White House Archives. January 30, 2016. [Online] Available: https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

[2] Google Inc. & Gallup Inc. "Trends in the State of Computer Science in U.S. K-12 Schools." 2016 [Online] Available: http://goo.gl/j291E0

[3] M. Guzdial and B. Morrison (2016, November). "Growing Computer Science Education Into a STEM Education Discipline," *Communications of the ACM*, vol. 59, no. 11, November 2016. [Online]. Available: https://cacm.acm.org/magazines/2016/11/209119-growing-computer-science-education-into-a-stem-education-discipline/fulltext

[4] E. Witherspoon, R. Higashi, C. Schunn, E. Baehr and R. Shoop. "Developing Computational Thinking through a Virtual Robotics Programming Curriculum". *ACM Transactions on Computing Education*, vol. *18 no.* 1, pp. 1-20, 2018. [Online]. Available: https://dl.acm.org/doi/10.1145/3104982

[5] E. Hamner, T. Lauwers, D. Bernstein, I. Nourbakhsh and C. DiSalvo. "Robot Diaries: Broadening Participation in the Computer Science Pipeline through Social Technical Exploration". *Association for the Advancement of Artificial Intelligence*. 2008.

[6] A. Liu, C. Schunn, J. Flot, and R. Shoop. "The role of physicality in rich programming environments". *Computer Science Education*, vol. *23* no. 4, pp. 315-331, 2013. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/08993408.2013.847165

[7] *Policy Brief: Education during COVID-19 and beyond*. (2020). The United Nations. [Online]. Available: https://www.un.org/development/desa/dspd/wp-content/uploads/sites/22/2020/08/sg_policy_brief_covid-19_and_education_august_2020.pdf

[8] A. Robins, J. Rountree and N. Rountree. "Learning and Teaching Programming: A Review and Discussion". *Computer Science Education*, vol. *13* no. 2, pp. 137-172, 2003.

[9] R. Pea. "Language-Independent Conceptual "Bugs" In Novice Programming". *Educational Computing Research*, vol. *2* no. 1, pp. 25-36, 1986.

[10] S. Grover, *Computer Science in K-12: An A-To-Z Handbook on Teaching Programming*. Edfinity, 2020.

[11] H. S. Hadani and S. Parvathy "What do we know about the expansion of K-12 computer science education?", August 2020. [Online]. https://www.brookings.edu/research/what-do-we-know-about-the-expansion-of-k-12-computer-science-education/

[12] M. Kölling, N. Brown, and A. Altadmri. "Frame-Based Editing: Easing the Transition from Blocks to Text-Based Programming". *WiPSCE '15: Proceedings of the Workshop in Primary and Secondary Computing Education*, pp. 29-38, *November 2015*.

[13] A. Sullivan and M. Bers. "Robotics in the early childhood classroom: learning outcomes from an 8-week robotics curriculum in pre-kindergarten through second grade". *International Journal of Technology and Design Education*, 2016. [Online]. Available: from https://ase.tufts.edu/DevTech/publications/robotics%20paper.pdf

[14] L. Margulieux and R. Catrambone. "Improving Programming Instruction with Subgoal Labeled Instructional Text". *Georgia State University Learning Sciences Faculty Publications*, 952-957, 2014.

[15] B. Morrison, L. Margulieux and A. Decker. "Using Subgoal Labeling in Teaching CS1". *SIGCSE '19: Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp. 1237, 2019.

[16] "Situating Constructionism". *Constructionism*. Ablex Publishing Corporation. 1991. [Online]. Available: https://web.media.mit.edu/~calla/web_comunidad/Reading-En/situating_constructionism.pdf

[17] J. Stockard, T. Wood, C. Coughlin and C. Khoury. "The Effectiveness of Direct Instruction Curricula: A Meta-Analysis of a Half Century of Research". *Review of Educational Research*, vol. *88* no. 4, pp. 479-507, 2018. [Online]. Available; https://journals.sagepub.com/doi/abs/10.3102/0034654317751919

[18] S. Puntambekar and R. Hubscher. "Tools for Scaffolding Students in a Complex Learning Environment: What Have We Gained and What Have We Missed?" *Educational Psychologist*, *40*(1). 2010. pp. 1-12. [Online]. Available: https://www.researchgate.net/publication/261619027_Tools_for_Scaffolding_Students_in_a_Complex_Learning_Environment_What_Have_We_Gained_and_What_Have_We_Missed

[19] E. Silk, R. Higashi, R. Shoop and C. Schunn. 'Designing technology activities that teach mathematics'. *The Technology Teacher*, vol. *69* no. 4, pp. 21-27, 2010. [Online]. Available: https://www.ri.cmu.edu/pub_files/2010/4/SilkEtal2010a-TTT.pdf

[20] D. Wiliam. "Formative Assessment: Getting the Focus Right". *Educational Assessment*, vol. *11*, pp. 283-289, 2006. [Online]. Available: https://www.researchgate.net/publication/248940867_Formative_Assessment_Getting_the_Focus_Right

[21] OECD/CERI International Conference. Assessment for Learning Formative Assessment. *Learning in the 21st Century: Research, Innovation and Policy*, 2008.