frontiers | Frontiers in Computer Science

TYPE Original Research
PUBLISHED 10 January 2023
DOI 10.3389/fcomp.2022.1031572

Check for updates

OPEN ACCESS

EDITED BY
José Lima,
Polytechnic Institute of Bragança (IPB), Portugal

REVIEWED BY
Maha Khemaja,
University of Sousse, Tunisia
Raffaele De Amicis,
Oregon State University, United States
Cucuk W. Budiyanto,
Sebelas Maret University, Indonesia

*CORRESPONDENCE
Gordon Stein
gordon.stein@vanderbilt.edu

SPECIALTY SECTION
This article was submitted to Digital Education, a section of the journal Frontiers in Computer Science

RECEIVED 30 August 2022
ACCEPTED 19 December 2022
PUBLISHED 10 January 2023

CITATION
Stein G, Jean D, Brady C and Lédeczi Á (2023) Browser-based simulation for novice-friendly classroom robotics. *Front. Comput. Sci.* 4:1031572. doi: 10.3389/fcomp.2022.1031572

COPYRIGHT
© 2023 Stein, Jean, Brady and Lédeczi. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

Browser-based simulation for novice-friendly classroom robotics

Gordon Stein[1]*, Devin Jean[1], Corey Brady[2] and Ákos Lédeczi[1]

[1]Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, United States, [2]Department of Teaching and Learning, Vanderbilt University, Nashville, TN, United States

Robots are a popular and engaging educational tool for teaching computational thinking, but they often have significant costs and limitations for classroom use. Switching to a simulated environment can eliminate many of these difficulties. By also providing students with a block-based programming environment, the barrier to entry can be further reduced. This paper presents a networked virtual robotics platform designed to create an environment which is highly accessible for novice students and their teachers alike, along with components of a curriculum designed to teach computational thinking skills through robotics programming challenges, including autonomous challenges and in-class competitions. Students access this platform through an extension of the same web interface used for programming their robots, which allows students to collaborate on code and view a shared simulated virtual space. Previously, this virtual robotics platform was used only to facilitate distance education. This paper demonstrates its use in an in-person class during the Spring 2022 semester, illustrating the affordances of a virtual robotics environment for face-to-face learning contexts as well. Students' computational thinking skills were evaluated with assessments both before and after the class, along with surveys and interviews given to determine their opinions and outlooks regarding computer science. The results show that students had a significant improvement in both attitudes and aptitudes.

KEYWORDS

educational robotics, robotics simulation, computational thinking, STEM education, distance education

1. Introduction

Demand for both STEM and distance education has continued to be strong for many years. Distance education had especially high demand during the COVID-19 pandemic (Crick et al., 2021). While students have now largely returned to in-person instruction, demand for distance education tools remains and is expected to continue (Siegel et al., 2021). This need to integrate the web in the *modality* of CS learning actually presents an opportunity to bring to the foreground internet-connected and distributed computing systems in the *content* of CS Education (Broll et al., 2021). As the number of Internet of Things devices continues to grow, students are expected to be equipped with relevant knowledge in computer science, distributed computing, and other related topics

(Abichandani et al., 2022). Educational tools must be provided to meet both of these needs, to grant educators the capability to reach students in any setting with STEM topics.

Educational robotics is a common tool to engage and excite students in STEM classes (Sapounidis and Alimisis, 2021). Robots give students tasks they can see represented in a physical world, while enabling both teamwork and competition, allowing a wide range of topics to be covered. This approach is also adaptable to suit a wide range of pedagogies, allowing for significant flexibility in curricular design. For example, many classrooms have used Lego robot kits, which use a block-based programming interface to make their functionality more accessible to novice programmers (Souza et al., 2018).

However, costs of robots for classroom use can be significant (Mistretta, 2022). In addition to the upfront cost of purchasing the robots themselves, there are costs associated with setting up the robots, maintaining the robots, and performing activities with the robots. Some robots may require assembly, most mobile robots require batteries to be replaced or charged before use, and, depending on the age and experience level of the students, any other repairs may be entirely the responsibility of the teacher, if the robots are user-repairable at all. This means teachers may be required to have additional electrical engineering knowledge, regardless of the topics they intend to use the robots with. Further, any activity performed with the robots will require some setup and cleanup, with extra care required to ensure equivalent starting conditions for in-class competitions. Internet-connected robots may require support from a school's IT department to allow connecting them to school networks, or they may require the educator to provide a wireless hotspot for them to connect to. Finally, the costs and technical difficulties of even simpler robots may prevent students from being able to take a robot home for homework or practice, reducing the time students get to work with them.

A common approach to eliminate these costs and difficulties is to switch to simulated robots (Homa, 2019; Tselegkaridis and Sapounidis, 2021). A simulated robot has no cost other than the price of the simulation software and the computer it runs on. Setting up activities for simulated robots only requires telling the program which scenario to use, and students may be able to install or access the software on their home computers to "take a robot home" with them without any risk that they will damage or lose school property. Simulated robots require no batteries, and "repairs" are often accomplished by simply resetting the simulation, making maintenance greatly simplified.

Computational Thinking (CT) (Wing, 2006) uses the cognitive tools associated with computer science extended to problem-solving in general. In general, it includes thinking algorithmically, framing problems so that technology can assist in a solution, organizing and analyzing data, abstracting to models and simulations, efficiently solving problems, and transferring the CT process to other domains (ISTE and CSTA, 2011; Weintrop et al., 2016). CT is presented as a preferred target for educators, as it not only is associated with STEM topics, but the "transfer" component inherently makes it applicable to a wide range of domains. Educational robots have been commonly used to teach CT (Chen et al., 2017; Chevalier et al., 2022), at every level, from kindergarten (Roussou and Rangoussi, 2020) to undergraduates in college (Aristawati et al., 2018).

A variety of approaches have been created to provide an alternative to physical educational robots or to provide access to robots for distance education. These have included both remote control and web-based simulation, each of which have been used for a significant length of time in various forms (Tzafestas, 2009). However, these approaches are not equivalent: providing remote access does not necessarily eliminate the same costs as simulation, nor does it necessarily provide a similar experience to typical physical educational robots. For example, Georgia Tech's "Robotarium" (Pickem et al., 2017) requires students to upload code to be verified in simulation first, with the goal of providing safe access to a shared robotics testbed. In contrast, our solution, RoboScape Online, focuses only on providing simulated spaces for students to use, eliminating any need to verify robot code will not damage the robots and enabling user input-based control.

On the robotics simulation side, multiple platforms have been created for the education space (Tselegkaridis and Sapounidis, 2021). One notable example is Gazebo (Open Source Robotics Foundation, 2022), which has already seen significant educational use (Gervais and Patrosio, 2022). Gazebo provides excellent physics simulation and can be configured for almost any realistic robot design, but its use in a classroom requires both a substantial hardware cost to run its simulations and/or significant technical knowledge. For instance, many solutions for simplifying the use of Gazebo in the classroom rely on creating a Docker container to run on each student computer. This approach may not be accessible to teachers with limited information technology experience, and there may be difficulties in obtaining approval from school IT departments or running on students' personal hardware at home. By performing the physics simulation remotely and providing a browser-based client for student use, our approach creates a more accessible system, while still achieving the performance and accuracy levels needed for educational scenarios.

Another notable preexisting robotics simulation platform is Robot Virtual Worlds (Robomatter Inc, 2022), which has also seen use in education research (Mistretta, 2022). This platform is much more straightforward for classroom use, but unlike Gazebo it is non-free, proprietary software, with support for Windows only, requiring the software to be installed before use. The VEX Virtual Robotics platform (VEX Robotics, 2022) and Robotify (Robotify, 2022) are two browser-based educational robotics simulation platforms also allowing the use of block-based programming environments. These systems are closed-source platforms with some content restricted behind payment.

In contrast, RoboScape Online aims to provide a free, open-source platform for educational robotics simulation in the browser, open to contributions from educators and students. While Robotify also allows for multiple users to interact in one networked environment, users have no ability to modify the program or host their own servers due to the closed-source nature of the platform. In addition, by building upon the foundation of NetsBlox's ecosystem (Stein and Lédeczi, 2021), RoboScape Online allows for more advanced robot programs, controls, and interactions to be created by students. For example, students can create a phone app that sends commands to the robot (Jean et al., 2021), or multiple students' programs can communicate over the network to synchronize or coordinate actions.

The work we report on here builds upon our previous work with physical robots. The RoboScape platform was used with physical robots and in-person classes to teach cybersecurity concepts to middle and high school level students (Lédeczi et al., 2019). In 2021, these classes were run as an online class using an earlier version of RoboScape Online (Stein and Lédeczi, 2022), created using the Unity game engine. Students remained engaged and positive about the experience after it moved to virtual robots. Our experience in implementing this earlier version supported the design decisions and architectural changes that produced the current version of Roboscape Online. In this article, we describe this system and present the results of implementing our next iteration of design-based research (Cobb et al., 2003) in Spring 2022.

2. Materials and methods

2.1. NetsBlox

The programming environment used in this work is NetsBlox Broll et al. (2017), a block-based environment which expands upon the Snap*!* (Mönig and Harvey, 2022) programming environment created at Berkeley. While Snap*!* is similar to Scratch (Maloney et al., 2010) in that both are visual programming environments designed to introduce computer science concepts through block-based abstractions, Snap*!* extends this concept by introducing ideas from the Scheme programming language to provide a more complete and powerful programming experience without sacrificing the novice-friendly block-based approach (Romagosa, 2019). This includes both the use of first-class, heterogeneous lists as the only structured data type and functional programming concepts such as custom block types and closures as first-class types. NetsBlox in turn adds an additional set of computer science concepts on top of Snap*!*, providing distributed computing features such as message passing and remote procedure calls, along with access to external web services (Broll et al., 2021).

These new features are largely made available through similar block-based abstractions to the ones already present in Snap*!*. For example, where Snap*!* has the ability to use messages to run events in other objects within the same program, NetsBlox adds blocks to send a message over the network to another instance of the same program, another "role" within the program, or another user's program, identified through an addressing system. This enables classroom challenges in which message types are used as an interface between a teacher's program and student programs, such as a chatroom where the students must all write client programs that interact with the server run by the teacher. Remote procedure calls (RPCs) are available as blocks that provide students with a list of available network services and methods they can call. This functionality is often used to provide a block-based abstraction for interacting with external web services such as Google Maps (see Figure 1), but it is also used to allow students to interact with robots, phones, and other internet-connected devices, or to get and set values from a shared cloud data store. These distributed computing features enable the creation of multiplayer video games, while also making it more feasible for students to interact with scientific data by integrating with both data sources and outputs such as charting software. By providing a novice-friendly interface for distributed computing, NetsBlox has supported multiple classes and camps aimed at high school and middle school students who have limited prior computer science experience.

The distributed computing features in NetsBlox go beyond those present in the blocks available for programming. A wide range of cloud-based functionality has been added to assist users. While unauthenticated users can only save documents to their browser's local storage, with a NetsBlox account, a user can save their projects to the NetsBlox cloud, providing both reliable storage and easy access across any computers they may be working on (e.g., classroom, library, home). NetsBlox projects can also be made public and shared using a specific link, allowing a user to easily give others access to their creations. Libraries of custom blocks can also be shared through the interface, facilitating the sharing of code snippets as well. NetsBlox also allows for students to collaborate remotely on a single project. Finally, these features are also integrated into a classroom system, giving teachers the ability to control over which other accounts their students are allowed to communicate with.

2.1.1. RoboScape

Robotics support in NetsBlox is provided through a service called RoboScape (Lédeczi et al., 2019). The RoboScape service exposes an interface to robots as if they were just another web service accessible through NetsBlox. For example, to drive a robot forward, a student could send the command "set speed 100 100" to the RoboScape service's "send" RPC to set both wheels to move forward at a specific speed. Figure 2 shows an example

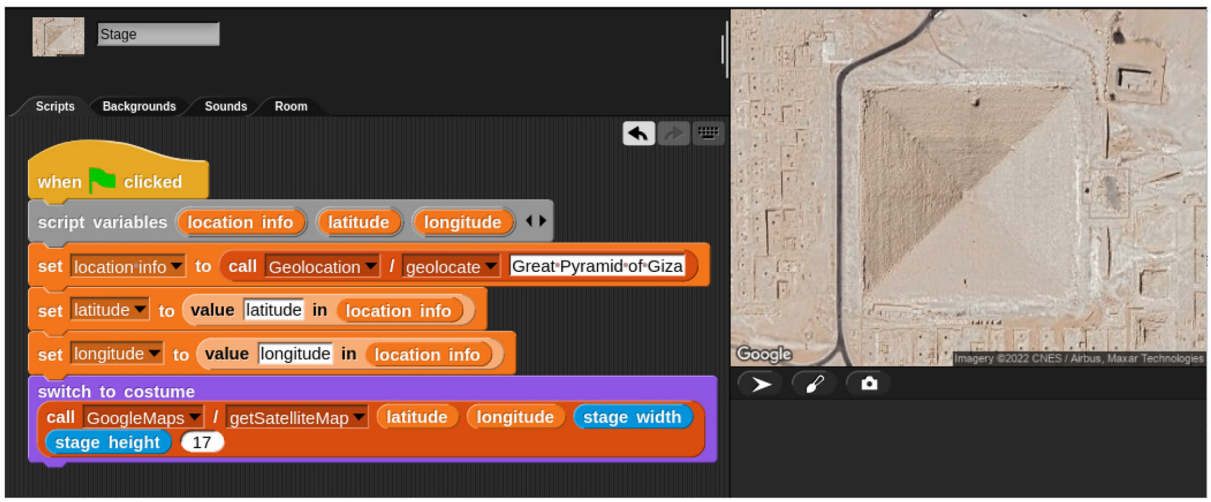

FIGURE 1
Example NetsBlox code, interacting with web services to search for a location and obtain a satellite view of it.

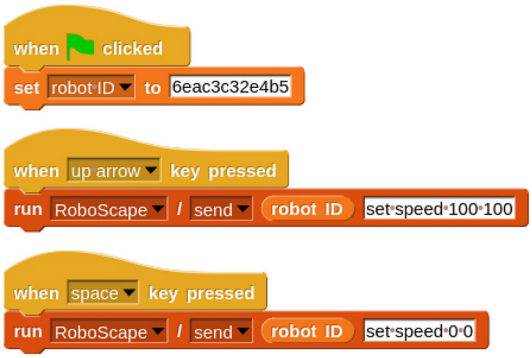

FIGURE 2
Simplified example RoboScape code. When the up arrow key is pressed, the robot moves forward. When the space bar is pressed, the robot will stop.

of NetsBlox code to move a robot forward or stop it based on user input. A student can also access other features of the robot, for example sending the command "get range" to receive the output value of a distance sensor on the robot as a response. This approach enables students to leverage their NetsBlox experience with other distributed computing concepts, to make a smooth conceptual transition into using robots. In addition, sending commands to robots in this way, as opposed to the traditional approach of writing code to be directly uploaded to the robots, creates the potential for novel learning opportunities with the robots as a tool.

In particular, as all robot commands are relayed through the NetsBlox server, it was possible to add features by which these commands can be intercepted, eavesdropped on, and spoofed for cybersecurity lessons. A previous curriculum using RoboScape during multiple camps and classes (Yett et al., 2020) focused on teaching students about concepts such as encryption, replay attacks, and denial-of-service attacks—along with defenses against such attacks—using features of the RoboScape service. As the robot code and the networking code on the server are provided for students, abstractions are easily afforded to allow students to interact with these concepts on a level that novices to computer science are more comfortable with. For example, a student wishing to protect their robot with encryption sends a "set key" command with their chosen key (or a "hardware key" obtained from the robot) to enable a Vigenre cipher that is easily implemented with blocks. In the cybersecurity curriculum, students engage in a series of competitions as they learn new concepts. The general structure involves one team attempting to complete a goal by maintaining control of their robot while the opposing team attempts to prevent them from completing their goal, using the cybersecurity topics covered. Students have responded well to the competitive environment, especially enjoying games such as a "tug-of-war" where two teams attempted to move a single robot to their side of the room.

2.2. RoboScape Online

The recent disruption of education by COVID-19 has motivated the creation of a platform that would allow access over the Internet to the learning opportunities of RoboScape. This has resulted in RoboScape Online, a simulation platform providing access to virtual versions of robots similar to those that were used with RoboScape during in-person classes.

The Parallax ActivityBot 360, with an attached daughterboard for wireless connectivity, as used for previous RoboScape activities, has a cost of approximately $280 without quantity or other discounts. While we have found these robots to be very reliable, they have more processing power and many more features than are necessary for use with RoboScape. Many of the features which are necessary, such as high-speed continuous servos with encoder output and Internet connectivity, are trivial for a virtual robot, but often costly in commercial-off-the-shelf educational robotics platforms customizable enough to add support for NetsBlox. Simulated robots eliminate the cost entirely while providing all the same features the physical robots had and more. Further, physical robots experienced much more latency and packet loss, making many autonomous challenges difficult for students to accomplish—for the wrong reasons. Simulating robots has additional logistical benefits as well, such as providing more consistent initial conditions without any effort expended for setup or cleanup, and not requiring batteries or repairs to continue functioning. Finally, RoboScape Online was specifically designed to not require any significant prior programming knowledge beyond that required to use NetsBlox, so that it would be accessible to a wider range of educators and students.

Providing students with shared virtual spaces has many benefits beyond costs. Students are able to collaborate in both their code and with their robots without having to be in immediate physical proximity. At the same time, through NetsBlox, robot-to-robot communication is made very simple. Easy resets to initial conditions and reliable connections assist students with debugging, and novel diagnostic tools are feasible with virtual robots to help students understand what their code is doing. The potential for competition with virtual robots is also enhanced. The shared virtual spaces are simulated on a remote server, providing a consistent simulation that does not rely on clients for anything but commands, and also reducing the system requirements for a computer to be compatible with the platform. A future version is planned to automate tournaments of multiple types, both for programs that require synchronous user input and fully automated tournaments of autonomous code with matches running concurrently.

With a simulated platform, it also becomes possible to provide sensors and actuators that would be infeasible for classroom use, due to either their cost or potentially hazardous nature, while making it easier to restrict the features available to students based on their assignments. For example, a robot in RoboScape Online can be equipped with a LIDAR sensor, with its range and angular resolution defined as part of the scenario the students are placed in. Additional virtual sensors add no cost and can be designed to add or eliminate effects such as noise to simplify them for student use. For instance, a virtual robot can be given a radiation sensor or a toxic gas sensor that detects the amount of hazardous emissions around

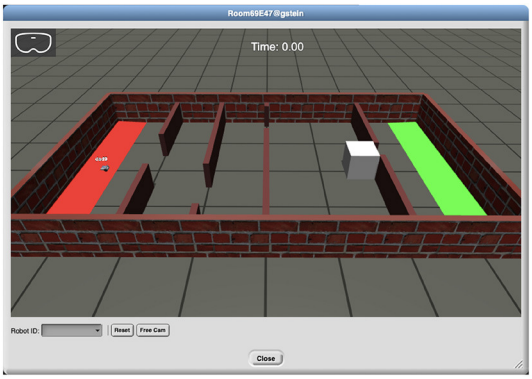

FIGURE 3
RoboScape Online 3D view showing "obstacle course" environment. The robot is in its starting position in the red area.

it: this would obviously be impossible to use with physical robots in a classroom, but it enables vivid scenarios (e.g., search and rescue) that students may find more interesting. These new sensors are made available to the programming interface through NetsBlox's IoTScape service (Tan et al., 2022), giving similar web service-like abstractions to what students already work with for the robots.

An earlier iteration of RoboScape Online was a standalone application created in the Unity game engine. This approach was chosen based on the ease of content creation, powerful rendering capabilities, and existing networking modules available. However, in practice, many limitations of this approach became apparent. To make full use of the content creation tools from Unity, the server-side simulation was required to be running a full Unity application running without graphics, which required an additional cloud server instance for each group of students using it to provide a consistent and performant simulation. This software worked well enough for initial, exploratory implementations in online classes with students using their own computers. However, moving back to the classroom exposed issues with this approach. The client software was able to be provided as binaries for multiple platforms, but installing software on school computers was often difficult to get permission for and difficult to update. Moreover, the networking system chosen required specific ports to be available on the client's network, which proved to be very difficult to negotiate with schools' IT departments. At the same time, it was observed that students were able to play online multiplayer games in the browser on school computers, inspiring the transition to an HTML5-based solution. The current iteration of RoboScape Online is a combination of a .NET server and a JavaScript client running through NetsBlox's extensions feature. This new approach is compatible with almost any computer that can run NetsBlox itself, requires no installation besides a Chromium-based browser, and it

uses a networking protocol which almost all networks should allow. Additionally, by providing the client as a NetsBlox extension, the 3D view is better integrated with the students' program development environment, preventing switching costs. Furthermore, new blocks can be added as part of the extension, such as a "robots in room" reporter block, to give students' code easier access to relevant robot IDs.

Many scenarios and environments are available for RoboScape Online. An early lesson in our curriculum has students write a custom remote control program to navigate through an obstacle course, as shown in Figure 3. Students may be given the task to have their robot autonomously navigate through a walled path by using a LIDAR sensor (see Figures 4, 5 for a simplified solution program), or to use a similar sensor to find boxes on a platform and push them off. A "treasure hunt" environment has students use a metal detector, which gives them readings of proximity to a large object; when their robot is close enough (the assignment is given both with manual and autonomous control), they must give it a command to "dig" to reveal a buried treasure, as shown in Figure 6. More open-ended environments give students a number of robots with a variety of capabilities, allowing them to design their own tasks.

2.2.1. Architecture

The server architecture for RoboScape Online is shown in Figure 7. In addition to the preexisting NetsBlox server, a main API server and multiple additional simulation servers were added. From the perspective of the NetsBlox server, the simulation is no different from any set of physical robots and additional sensors. The virtual robots communicate with the server with the same protocol as the physical robots, which has allowed the existing RoboScape code to be reused with RoboScape Online. The NetsBlox server also provides storage for student project data, communication between student projects, and collaboration features. Students creating or joining rooms communicate with the main API server over HTTP requests. When a simulation server is started, it announces itself, advertising its supported scenarios and its capacity to the main API server. A student's request for a new room will be redirected to the compatible server with the greatest remaining capacity. Requests to join existing rooms will be redirected to the simulation server hosting that specific room, if it exists. Once redirected to a simulation server, the client opens a WebSocket connection, over which updates to the state of the simulation from the server are streamed to the client, and commands from the client are sent to the simulation server. When no users are in a room or no activity has occurred for several minutes, the room is put into "hibernation", allowing users to rejoin to resume the simulation at the state it was left in. Hibernating rooms are removed if they are not rejoined after several days.

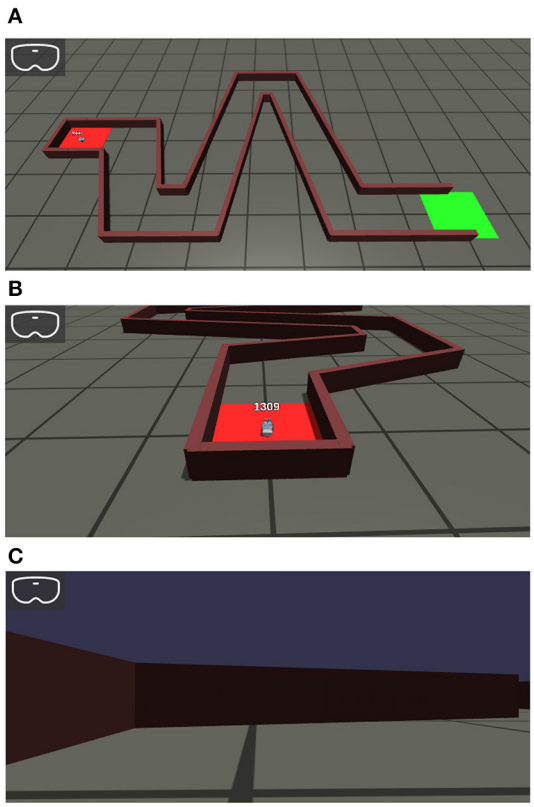

FIGURE 4
RoboScape Online 3D view showing "LIDAR road" environment. The robot is in its starting position in the red area. **(A)** Overview of course. **(B)** "Chase cam" view of robot. **(C)** "First person" view of robot.

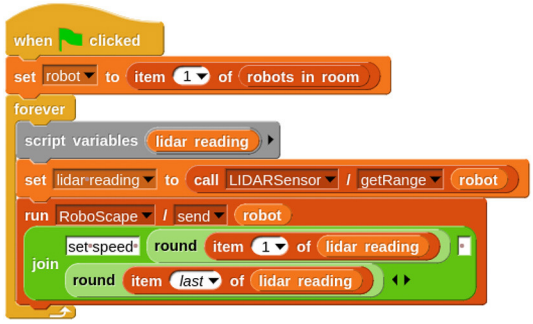

FIGURE 5
Simple solution to "LIDAR road" challenge.

2.3. Curriculum

A curriculum focusing on robotics programming tasks was created for use in design-based research to support and study the classroom use of RoboScape Online as an

educational tool for teaching CT. The course took place over 7 weeks in the Spring 2022 semester, during which the class met to engage with RoboScape Online once a week for 2 h. Activities included an initial introduction to NetsBlox and distributed computing, followed by challenges that involved creating a manual remote control program, building autonomous driving programs to navigate environments and solve tasks using a LIDAR, coordinating waypoint navigation with a simulated GPS, and more. Students worked in groups using the collaboration tools in NetsBlox and the ability to share a single RoboScape Online room.

Competitions were used in the classroom to motivate and engage students. The scenarios used all had a timer feature included, allowing a simple measure of the performance of student code. After the competitions ended, students, especially the groups with the best performing programs, were invited to explain how their code worked and to share their strategies and lessons learned with their classmates.

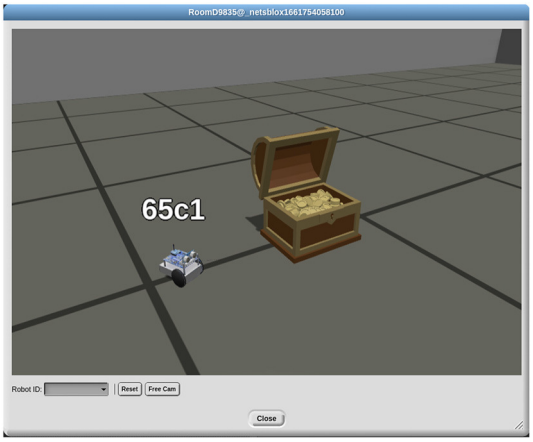

FIGURE 6
Visual of success state in "treasure hunt" environment. The treasure chest was not revealed until the robot moved to its location and a "dig" command was sent to it.

2.3.1. Participants

The class of students who participated in this implementation consisted of 27 high-school students from the School for Science and Math at Vanderbilt (Eeds et al., 2014). Of the 16 students who responded to both the pre and post surveys, nine identified as male and seven as female. Four students identified as white, three as Asian, six as black, one as American Indian or Alaskan Native, and two as another race.

Over 70% of students claimed to be familiar with a block-based programming language previously (answering "Agree" or "Strongly Agree" to the question "I have used block-based programming like Scratch before"), with 43% claiming the same for a text-based language.

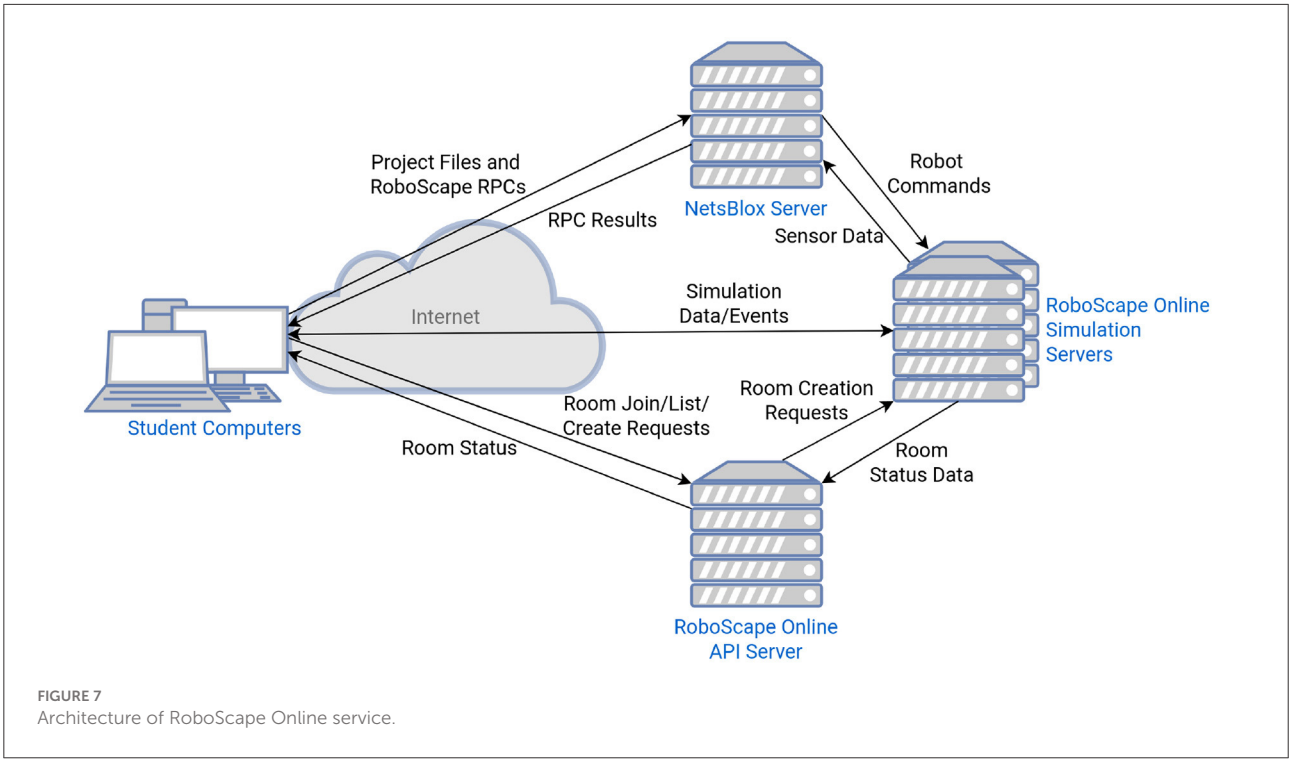

FIGURE 7
Architecture of RoboScape Online service.

2.3.2. Instruments

Student data was collected using three different instruments: surveys, tests, and interviews. A survey and test were given both before and after the implementation. Additionally, a group of students were selected for interviews after the final session of the activity sequence.

2.3.2.1. Surveys

Student demographics and opinions were collected through an online survey administered to students both before and after the course. Each opinion question was presented to students on a four-point scale, ranging from "Strongly Disagree" at 1, to "Strongly Agree" at 4. Survey questions were designed to identify the impact of the implementation on student attitudes and beliefs about robotics and programming, which were conjectured to mediate their changes in programming performance. As illustrated in Table 2, questions aimed to capture students' confidence and self-efficacy around programming and robotics. These are similar to the questions used in Cross et al. (2016) to evaluate student confidence with educational robotics.

2.3.2.2. Pretest/Posttest

To evaluate the educational effect of participating in the course, students were given an in-class programming assessment both at the start of the first day and end of the last day of the course. Both the pretest and posttest consisted of a pair of programming tasks that students were expected to accomplish using a provided pseudocode language. While some aspects of the language, such as the presence of "repeat until" loops, were designed to resemble NetsBlox, some aspects of it differed, including greatly simplified commands for interacting with robots. One question was shared between both tests (question 2 on the pretest, question 1 on the posttest) to provide a baseline. The tests' tasks focused on robotics programming tasks; however, to measure the students' ability to transfer their knowledge beyond the contexts presented in the curriculum, the specific topics differ significantly from the contents of the course.

We applied a rubric and grading process informed by the work of Chen et al. (2017). Their approach considers students' code through a five-component framework. Though we omitted the "Representation" and "Data" components of their framework, we used the "Syntax," "Algorithm," and "Efficiency" components. For each assessment item, each component was scored from 0 to 2, guided by the rubric shown in Table 1. Two researchers graded half of the students' responses and then evaluated inter-rater reliability for each component through percentage of agreement. If the agreement percentage was >85% for all components, the graders would then continue on to evaluate the remaining responses. If not, the graders met to discuss their disagreements until the desired inter-rater reliability was reached. This process was then repeated for the remaining responses.

TABLE 1 Rubric used to evaluate student responses on pretest and posttest.

Syntax	2	Commands in solution obey given syntax almost totally
	1	Commands in solution obey given syntax partially
	0	Commands in solution do not obey given syntax at all or are not psuedocode
Algorithm	2	Solution solves the problem
	1	Solution solves the problem partially or displays good understanding of problem/solution form
	0	Solution does not solve the problem and does not display understanding of problem/solution form
Efficiency	2	Solution has no or almost no redundant or unnecessary code
	1	Solution has some redundant or unnecessary code
	0	Solution is mostly redundant or unnecessary code

2.3.2.3. Interviews

To augment the survey information and provide richer context for some of these effects, semi-structured interviews were conducted with seven participants at the end of the unit. These interviews ranged in length from 7 to 32 min, and were conducted by the authors.

In analyzing these data, the first author repeatedly read the transcripts as well as the viewed video recordings of the interviews. This familiarization process was guided by constant-comparative methods (Glaser, 1965) and inductive coding (Strauss and Corbin, 1990; Charmaz, 2006). The generation of codes was not entirely an effort of grounded theory (Glaser and Strauss, 1967), however, as the literature and the survey results provided important sensitizing concepts (Blumer, 1954).

During this process, the first author identified themes across the participants' experiences, which served both to provide depth of description and as a means of triangulating (Creswell and Miller, 2000) these qualitative data with the quantitative survey results.

3. Results

3.1. Surveys

Due to the limited number of survey responses, only three questions reached the threshold of statistical significance. These questions are listed in Table 2. In general, the students responses to the open-ended questions on the post-survey were positive, with some examples listed in Table 3. These three responses suggest that students' gained confidence in their general ability to engage with CS and programming; with their ability to create personally and socially meaningful constructions with code (i.e., apps); and with their understanding of the logic involved in defining robot behavior. For the question "I can program

TABLE 2 Significant pre-survey and post-survey differences.

Question	Pre-survey average	Post-survey average	Effect size (Cohen's d)	p-value
I am able to do well in activities that involve programming and computer science.	2.688	3.063	0.687	<0.01
I can program computers to create new apps (in other words, write code).	1.800	2.188	0.470	<0.03
I can explain how a robot makes decisions.	2.625	3.125	0.655	<0.004

TABLE 3 Post-survey open ended questions and selected responses.

Question	Responses
What would you tell a friend about this course?	"I really enjoyed it and further reinforced my interest in coding and robotics engineering." "This class was extremely interesting and being able to solve problems was always rewarding."
What did you like about the virtual environment?	"It was convenient, it was extremely portable in comparison to real-life robots and it was free (or at least for my class) It was also relatively easy to understand the code blocks and what they did." "It worked a little more reliably than a robot would in real life." "I liked that we could reset them in a very easy and efficient way."
What was your favorite activity?	"I really enjoyed coding the LIDAR navigation program." "I really enjoyed them all, but if I have to choose one, I would say the very first autonomous robot activity, which dealt with the robot(s) pushing the boxes off the 'hovering white floor'. I like it because it was the very first time I was programming a virtual robot to autonomously do a task." "I liked coding as homework and then coming back to class were we would test our creations and get feedback."
What were some (one or more) takeaways, "a-ha"s, or insights from this program?	"Like in math there is always a way to solve something but some things are more difficult than others." "Programming can be used for a lot more that just coding apps, robots, etc. " "I learned the aspect of how things virtually are made with the concept of coding. Also, an 'a-ha' for me was when I learned how deep/complicated coding could get with so much code."

computers to create new apps," the pre-survey student response was <2, meaning that the average student disagreed that they had this ability. After the course, the average score was now >2, indicating that students' perception of their abilities was now such that they feel confident enough to report computer programming as an ability.

3.2. Pretest/Posttest

Aggregate scores for the graded pretests and posttests are presented in Table 4. The changes from pretest to posttest for the Syntax and Efficiency categories were not statistically significant, but the change in the Algorithm component was ($p < 0.02$), with an effect size (Cohen's d) of 0.469. Next, student scores were separated based on previous experience with block-based programming languages to determine the impact of prior experience on their performance. While students with past experience had higher scores on average for each component on both the pretests and posttests, this difference was not statistically significant ($p > 0.05$) except on the Efficiency component of the posttest, where experienced students had an average score of 1.66 and non-experienced students had an average score of 1.14.

TABLE 4 Pre-test and post-test results, significant ($p < 0.05$) column marked with asterisk.

Question	Syntax	Algorithm*	Efficiency
Pretest 1	1.750	0.708	1.396
Pretest 2	1.813	0.646	1.063
Posttest 1	1.750	0.729	1.170
Posttest 2	1.667	1.167	1.458
Pretest combined	1.782	0.677	1.229
Posttest combined	1.708	0.948	1.313

All scores have a maximum value of 2.

In spite of improvements in many areas, our tests revealed some students had persistent conceptual difficulties across both tests. Multiple students demonstrated misunderstandings of how variables worked. Some seemed to be treating variables as if they were preprocessor macros, e.g., using "set right to get color right" and expecting the value of "right" to always be the value of running "get color right," while a few others seemed to think that an assignment "set right to get color right" would mean that a statement of only "get color right" updates the variable's value without a proper assignment statement. Some students repeated initial lines verbatim from an example program they were given, which performed no function in their new program, even if they otherwise understood how to solve the problem. This negatively effected their Efficiency scores, but as the copied lines of pseudocode were valid and had no effect, their Syntax and Algorithm scores were not impacted. These challenges indicate potential directions for refinements to the curriculum and facilitation of the course.

3.3. Interviews

All students interviewed commented on their enjoyment of problem solving associated with the tasks during the class, and many said they found it especially satisfying to complete a task. For instance, one remarked that the open-ended nature of the challenges were "really like a large puzzle and I feel like that was the best part about it." Another student specifically mentioned feeling that the course had improved their critical thinking skills (likely referring to the algorithmic component of computational thinking skills), saying "now when I evaluate problems and just look at them and analyze them, I can think through them clearer." Finally, another student mentioned enjoying the low-risk nature of the in-class competitions, saying, "You don't really lose anything if you do lose, so I think that helped."

In general, the students felt that the virtual robots were simpler than physical ones (whether speaking hypothetically or from their own experience). One student noted the advantage to debugging with a virtual robot, saying, "If you have a physical robot, you don't know if it's the code, or it could be the physical robot, it could bump into something, maybe one of the motors isn't working, or one of the treads is not working, maybe something's a little bit loose." Two students noted the low-cost aspect of simulation; one said, "I think it's really good in the sense that you don't have to actually buy a robot or something and you can actually work on a robot, but virtually. And so I really like that experience. Also it's easier to work with your code and the robot and stuff." The other remarked, "I like how virtual you can do it anywhere, and this is free, not to buy it." Finally, students also mentioned that they enjoyed the block-based development environment; as one said, "I think blocks are easier because you know what the function of the block is. Just because you can put them together, how the blocks fit together I felt makes a lot of sense."

In contrast, the students who had prior experience felt there were advantages to the physical robots, although they had some difficulty expressing the exact differences they experienced. One said, "It's easier to wrap your mind around what I do logically when you have it in front of you, as opposed to online." Another student recommended the use of physical robots "just to begin, to understand the actual concepts" and then suggested shifting to virtual robots for more advanced sensors. All students commented on the technical difficulties that emerged during the implementation, although the general attitude was that this was a part of working with experimental software. One said, "Sometimes I had technical difficulties, that was the only negative, but that's out of your control, so that's nothing wrong."

Students also expressed an increased or continuing interest in computer science classes after the robotics curriculum. All students interviewed expressed an interest in more computer science courses in their high school programs. Some expressed a desire for their schools to have more offerings in the field; as one said, their schools "have a lot of arts and performing arts but no CS," while another remarked that "if there was a [programming club at their school] I would think about joining it."

4. Discussion

This design-based research study implemented the new design and architecture of RoboScape Online, to support and study virtual robotics learning among early high school students. The results help to demonstrate the potential of curriculum using simulated robotics for in-person learning as well as in hybrid and remote classrooms. The study confirmed key design propositions of RoboScape Online—including presenting robots' functionality through the RPC-style interfaces characteristic of NetsBlox's approach to distributed computing. And it validated the technical feasibility of the new architecture. These findings motivate future work to implement the RoboScape Online platform and technologies at scale.

Additionally, the curriculum used in this study produced results that are encouraging, even in light of the limitations that the nature of our participant group impose on their immediate generalizability. Participating students seem to have finished the course with both an improved ability for computational thinking and, as indicated by results from the surveys and interviews, greater confidence in these abilities. While their pre-to-post test scores for Syntax and Efficiency did not change significantly, these were not the focus of the curriculum, and many students' Syntax pretest scores were already at or near the upper limit of the assessment's scale. Examining these findings at a finer grain-size will help us to refine our assessments for future implementation cycles. In particular, it is possible that the pseudocode language we used for the tests was simple enough that students rarely made enough of an error to reduce their Syntax score significantly; alternatively, the amount of prior exposure to programming that our participants had gave them sufficient familiarity with syntax in general to demonstrate a high level of fluency.

On the other hand, the students' gains in the Algorithm component were highly encouraging. This is the focus of the RoboScape Online curriculum and of our work with NetsBlox more generally. Students' increased proficiency and confidence in this area suggest important gains in Computational Thinking that can make a compelling case for including virtual robotics in middle-school and high-school students' experiences.

4.1. Limitations and future work

This work demonstrates the utility of the RoboScape Online platform in a classroom setting at the scale of a pilot project. In future work, larger-scale studies involving more classrooms and schools will provide a larger sample size and allow for

exploration of additional variables alongside the use of the virtual robotics platform.

Several limitations of the present study should be explicitly mentioned. Due to the research team's limited (once-a-week) access to students during the semester, the effect attributed to this course could be confounded with students' learning in other classes or with other factors more broadly. These could not be fully controlled or accounted for in our study. That said, the students involved were not taking any other coursework intentionally focusing on CT. Moreover, students' feedback in interviews and open-ended survey questions referring to the course specifically suggested its positive influence, but these provide only subjective metrics. Nonetheless, future studies would benefit from utilizing students not participating in the CT class as a control.

As another form of control and comparison group, it would be illuminating to study a second class using physical robots and a minimally-adapted RoboScape curriculum. Both RoboScape and RoboScape Online have strong claims to provide good outcomes for students, but a direct comparison of the two would be helpful for educators deciding between physical and virtual robots for their classrooms. In such a study, it would also be possible to explore the qualitative differences in learning and reasoning that can emerge across physical and virtual settings. Building upon comments from interviews with students who had experienced physical robotics in the past, there is reason to believe that there are differences and also that these differences are challenging for students to articulate. Findings from such a comparison study could be used to enrich the future development of both RoboScape and RoboScape Online. In particular, we have open questions about how to integrate principles and ideas from electrical and mechanical engineering. This is an important aspect of educational robotics which should be addressed in developing future extensions of both platforms.

A final limitation of our participant group is that the students had a relatively high level of prior experience with CT topics and programming in general. Some of the students had taken related courses in the past. While there were important advantages to implementing the experimental version of RoboScape Online with this group, future studies should engage with students who have less prior knowledge. A class of less-experienced students might demonstrate a greater change in the Syntax and Efficiency components between the pretest and posttest. Alternatively, if the Syntax component continues to show little change, it may be useful to replace the pseudocode language used with a subset of or variation on the block-based language used during the course itself to allow for automated grading of student responses. And it would be illuminating to determine whether the key gains in the Algorithm component could be sustained with a group of students with less experience, although our prior work with NetsBlox gives us reason to expect this is possible.

5. Conclusion

Robotics is an engaging topic, presenting opportunities for novice programmers to encounter ideas that are emerging in importance for computer scientists, STEM professionals, and citizens in general. Moreover, with the NetsBlox approach to RoboScape and RoboScape Online, these ideas in robotics are contextualized as vivid examples of distributed computing concepts. Though COVID-19 accelerated research to virtualize the experience of educational robotics, the present study suggests that virtual robotics can be a powerful addition to in-person learning as well, making advanced computing and engineering problems accessible to students with a wider range of experience and to lower-resourced schools. This study indicates rich directions for future research to make Computational Thinking attractive and accessible to a broader set of students at earlier stages in their learning.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Ethics statement

The studies involving human participants were reviewed and approved by Vanderbilt University Institutional Review Board. Written informed consent from the participants' legal guardian/next of kin was not required to participate in this study in accordance with the national legislation and the institutional requirements.

Author contributions

GS contributed most code for the RoboScape Online software used, performed the statistical analysis, and wrote the first draft of the manuscript. All authors were involved in data collection, wrote sections of the manuscript, contributed to conception and design of the study, manuscript revision, read, and approved the submitted version.

Funding

This material is based upon work supported by the National Science Foundation under Grant No. 1835874, the National Security Agency (H98230-18-D-0010), and the Computational Thinking and Learning Initiative of Vanderbilt University.

Acknowledgments

The authors would like to thank the School for Science and Math at Vanderbilt, a collaborative endeavor of Vanderbilt University and Metropolitan Nashville Public Schools (MNPS). We are also grateful to Dr. Brian Broll for the continued development and maintenance of the NetsBlox environment and to Dr. Shuchi Grover for assisting in the design of the survey instrument.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Author disclaimer

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Abichandani, P., Sivakumar, V., Lobo, D., Iaboni, C., and Shekhar, P. (2022). Internet-of-things curriculum, pedagogy, and assessment for STEM education: a review of literature. *IEEE Access*. 10, 38351–38369. doi: 10.1109/ACCESS.2022.3164709

Aristawati, F., Budiyanto, C., and Yuana, R. (2018). Adopting educational robotics to enhance undergraduate students' self-efficacy levels of computational thinking. *J. Turkish Sci. Educ*. 15, 42–50. doi: 10.12973/tused.10255a)

Blumer, H. (1954). What is wrong with social theory? *Am. Sociol. Rev*. 19, 3–10. doi: 10.2307/2088165

Broll, B., Lédeczi, A., Volgyesi, P., Sallai, J., Maroti, M., Carrillo, A., et al. (2017). A visual programming environment for learning distributed programming, in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (ACM), p. 81–86. doi: 10.1145/3017680.3017741

Broll, B., Lédeczi, K., Stein, G., Jean, D., Brady, C., Grover, S., et al. (2021). "Removing the walls around visual educational programming environments," in *2021 IEEE Symposium on Visual Languages and Human-Centric Computing* (IEEE). p. 1–9. doi: 10.1109/VL/HCC51201.2021.9576399

Charmaz, K. (2006). *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. New York, NY: Sage.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., and Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Comp. Educ*. 109, 162–175. doi: 10.1016/j.compedu.2017.03.001

Chevalier, M., El-Hamamsy, L., Giang, C., Bruno, B., and Mondada, F. (2022). "Teachers' perspective on fostering computational thinking through educational robotics," in *Robotics in Education*, Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R., and Obdržálek,, D., editors. p. 177–185, Cham: Springer International Publishing. doi: 10.1007/978-3-030-82544-7_17

Cobb, P., Confrey, J., diSessa, A., Lehrer, R., and Schauble, L. (2003). Design experiments in educational research. *Edu. Res*. 32, 9–13. doi: 10.3102/0013189X032001009

Creswell, J. W., and Miller, D. L. (2000). Determining validity in qualitative inquiry. *Theory Pract*. 39, 124–130. doi: 10.1207/s15430421tip3903_2

Crick, T., Knight, C., Watermeyer, R., and Goodall, J. (2021). "The International Impact of COVID-19 and "Emergency Remote Teaching" on Computer Science Education Practitioners," in *2021 IEEE Global Engineering Education Conference* (IEEE), p. 1048–1055. doi: 10.1109/EDUCON46332.2021.9453846

Cross, J., Hamner, E., Zito, L., Nourbakhshh, I., and Bernstein, D. (2016). "Development of an assessment for measuring middle school student attitudes towards robotics activities," in *2016 IEEE Frontiers in Education Conference* (IEEE), p. 1–8. doi: 10.1109/FIE.2016.7757677

Eeds, A., Vanags, C., Creamer, J., Loveless, M., Dixon, A., Sperling, H., et al. (2014). The school for science and math at vanderbilt: an innovative research-based program for high school students. *Life Sciences Educ*. 13, 297–310. doi: 10.1187/cbe.13-05-0103

Gervais, O., and Patrosio, T. (2022). "Developing an introduction to ROS and Gazebo through the LEGO SPIKE Prime," in *Robotics in Education*, Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R., and Obdržálek, , D., editors. Cham: Springer International Publishing. p. 201–209. doi: 10.1007/978-3-030-82544-7_19

Glaser, B., and Strauss, A. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Lippincott-Raven. doi: 10.1097/00006199-196807000-00014

Glaser, B. G. (1965). The constant comparative method of qualitative analysis. *Social Prob*. 12, 436–445. doi: 10.1525/sp.1965.12.4.03a00070

Homa, A. I. R. (2019). Robotics Simulators in STEM education. *Acta Scientiae*. 21, 178–191. doi: 10.17648/acta.scientiae.5417

ISTE and CSTA (2011). *Operational Definition of Computational Thinking*.

Jean, D., Broll, B., Stein, G., and Lédeczi, A. (2021). "Your phone as a sensor: Making IoT accessible for novice programmers," in *2021 IEEE Frontiers in Education Conference* (IEEE), p. 1–5. doi: 10.1109/FIE49875.2021.9637272

Lédeczi, A., Maroti, M., Zare, H., Yett, B., Hutchins, N., Broll, B., et al. (2019). "Teaching cybersecurity with networked robots," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (ACM). p. 885–891. doi: 10.1145/3287324.3287450

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. (2010). The Scratch programming language and environment. *ACM Trans. Comput. Educ*. 10, 4. doi: 10.1145/1868358.1868363

Mistretta, S. (2022). "Virtual robotics in hybrid teaching and learning," in *New Updates in E-Learning*. London, UK: IntechOpen. doi: 10.5772/intechopen.102038

Mönig, J., and Harvey, B. (2022). Snap! Available online at: https://snap.berkeley.edu/. (accessed March 8, 2021).

Open Source Robotics Foundation (2022). *Gazebo*. Available online at: http://gazebosim.org/ (accessed August 30, 2022).

Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., and Egerstedt, M. (2017). "The Robotarium: A Remotely Accessible Swarm Robotics Research Testbed," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. p. 1699–1706. doi: 10.1109/ICRA.2017.7989200

Robomatter Inc (2022). *Robot Virtual Worlds*. Available online at: https://www.robotvirtualworlds.com/ (accessed August 30, 2022).

Robotify (2022). *Robotify*. Available online at: https://www.robotify.com/ (accessed August 30, 2022).

Romagosa, B. (2019). *The Snap! Programming System*. p. 1–10. Cham: Springer International Publishing. doi: 10.1007/978-3-319-60013-0_28-2

Roussou, E., and Rangoussi, M. (2020). "On the use of robotics for the development of computational thinking in kindergarten: Educational intervention and evaluation," in *Robotics in Education*. Merdan, M., Lepuschitz, W., Koppensteiner, G., Balogh, R., and Obdrvzálek, D., editors, p. 31–44, Cham. Springer International Publishing. doi: 10.1007/978-3-030-26945-6_3

Sapounidis, T., and Alimisis, D. (2021). Educational robotics curricula: current trends and shortcomings. *Stud. Comp. Intellig.* 982, 127–138. doi: 10.1007/978-3-030-77022-8_12

Siegel, A. A., Zarb, M., Alshaigy, B., Blanchard, J., Crick, T., Glassey, R., et al. (2021). "Teaching through a Global Pandemic: Educational Landscapes Before, During and After COVID-19," in *Proceedings of the 2021 Working Group Reports on Innovation and Technology in Computer Science Education* (ACM), p. 1–25. doi: 10.1145/3502870.3506565

Souza, I. M. L., Andrade, W. L., Sampaio, L. M. R., and Araujo, A. L. S. O. (2018). A Systematic Review on the use of LEGORobotics in Education, in *2018 IEEE Frontiers in Education Conference (FIE)*. p. 1–9. doi: 10.1109/FIE.2018.8658751

Stein, G., and Lédeczi, K. (2021). Enabling collaborative distance robotics education for novice programmers, in *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. p. 1–5. doi: 10.1109/VL/HCC51201.2021.9576314

Stein, G., and Lédeczi, A. (2022). Shared virtual worlds for accessible classroom robotics, in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2, SIGCSE 2022*. New York, NY, USA: Association for Computing Machinery. p. 1177. doi: 10.1145/3478432.3499259

Strauss, A., and Corbin, J. (1990). *Basics of Qualitative Research*. New York: Sage.

Tan, Y., Rizk, M., Stein, G., and Lédeczi, A. (2022). "User-extensible block-based interfaces for internet of things devices as new educational tools," in *SoutheastCon 2022*. p. 711–717. doi: 10.1109/SoutheastCon48659.2022.9763937

Tselegkaridis, S., and Sapounidis, T. (2021). Simulators in educational robotics: a review. *Edu. Sci*. 11, 11. doi: 10.3390/educsci11010011

Tzafestas, S. G. (2009). *Web-based control and robotics education*. Springer: Intelligent Systems, Control and Automation: Science and Engineering. doi: 10.1007/978-90-481-2505-0

VEX Robotics (2022). *VEXcode Virtual Robotics (VR)*. Available online at: https://www.vexrobotics.com/vexcode/vr (accessed August 30, 2022).

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., and Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *J. Sci. Edu. Techn*. 25, 127–147. doi: 10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. *Commun. ACM*. 49, 33–35. doi: 10.1145/1118178.1118215

Yett, B., Hutchins, N., Stein, G., Zare, H., Snyder, C., Biswas, G., Metelko, M., and Ledeczi, A. (2020). A hands-on cybersecurity curriculum using a robotics platform, in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20. p. 1040–1046, New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3328778.3366878