Enabling Collaborative Distance Robotics Education for Novice Programmers

Gordon Stein
Institute for Software Integrated Studies
Vanderbilt University
Nashville, Tennessee 37212–2328
Email: gordon.stein@vanderbilt.edu

Ákos Lédeczi
Institute for Software Integrated Studies
Vanderbilt University
Nashville, Tennessee 37212–2328
Email: akos.ledeczi@vanderbilt.edu

***Abstract*—Distance education has gained significance recently. However, its application to robotics education presents challenges as physical access to hardware is typically required. While educational robotics simulation platforms exist, most are limited in scope or do not facilitate remote collaboration well. This paper proposes a novel networked robotics simulator for education, where students are able to collaborate in a shared 3D virtual space their robots inhabit. The framework combines these simulated worlds with a block-based programming environment to enable distance robotics education for a wider audience. The users' programs run in a web browser on their computers and they issue commands to the virtual robots through a network protocol abstracted to simple-to-use blocks. Robots send back acknowledgements and sensor values through the same protocol. Interactive environments provide students with a more immersive educational experience, and allow for automatically evaluating student performance. In addition to sharing the virtual worlds remotely, students can also develop their programs together since the block-based environment supports both synchronous and asynchronous remote collaboration. The paper presents two scenarios showing a robotics challenge and an extension of the concept to a smart city scenario controlling traffic lights. Reduced barriers to entry for both robotics education and curriculum creation will allow for a more diverse set of students and course materials.**

I. Introduction

While the COVID-19 pandemic created a more urgent need for students to engage in learning experiences without needing to physically enter classrooms, a demand for distance education in all fields has been present for decades [1]. While distance learning has provided many challenges in how students and educators both adapt to it, digital transformation of instructional activities can also provide new opportunities for richer educational experiences and provide flexibility to students who would otherwise be unable to enter the classroom.

Educational robotics has been demonstrated to enhance student interest and engagement, while encouraging underrepresented groups' participation in STEM [2]. However, classroom use of educational robotics requires the physical presence of robots and students and presents educators with several barriers to entry. The first, most obvious, cost is the inherent expense in purchasing the robots. Robots will also require maintenance to remain in use, and educators may need to be trained to repair the robots if damage occurs in the classroom. Challenges given to students may also include costs associated with setting them up or cleaning up afterwards. By using a simulation of robots, these costs can be greatly reduced. At the same time, physical robots in a classroom are limited in the range of scenarios presentable by them. Some robot designs, such as UAVs, may be difficult to utilize safely in schools, but a simulated robot ensures the students' safety.

A reduction in cost creates a more equitable setting for both students and educators, who may otherwise find themselves excluded from existing educational robotics platforms. In addition, by allowing for a wider range of curricula, including educational content created by instructors to personalize the content for their class, a more diverse group of student interests can be met.

A. Existing Work

Educational robotics has been present in K-12 classrooms for decades, with a variety of platforms being popular in education [3] [2]. Educational robotics simulators have also seen common use in classrooms [4] [5]. However, simulation for robotics education often has a different focus than this work. For example, Gazebo [6] provides excellent physics and sensor simulation, and provides some support for networking, but requires experience with Linux to set up, creating a significant barrier to entry for K-12 educators. A goal of this work, RoboScape Online, is to make it easy for educators to begin using virtual robots in their classrooms without requiring teachers be familiar with any technology outside of the familiar block-based programming interface used, NetsBlox, and a web browser. One notable educational robotics platform is Robotify [7], which also features block-based programming and multi-user collaboration. RoboScape Online will provide a free service with open source software designed to be extended with new content by educators, and even by students, compared to Robotify's proprietary for-profit service.

A solution to students being required to be physically present in a classroom is to implement "Robots as a Service", exposing robots as web services [8], similar to our original RoboScape service we are extending to virtual robots in this work, to enable remote-control over a greater distance.

978-1-6654-4592-4/21/$31.00 ©2021 IEEE

Although the system described by Wang et al. has the ability to connect to a Unity-based simulation, it is not focused on driving student collaboration through the tool or expanding it beyond college-level EE courses.

A similar approach to providing remote access to small robots for education is Georgia Tech's "Robotarium" [9]. However, in the Robotarium, students upload code to be verified in simulation first, with a focus on providing safe multi-user sharing of a physical testbed. RoboScape Online instead targets real-time simulation of robots in virtual testbeds—a large number of which can coexist simultaneously—allowing for greater scalability without significant costs, and reducing any need to ensure user code will not damage the robots, as simulated robots can be replaced with the click of a button.

Our previous approach, using physical robots with the RoboScape platform, was developed for summer programs run at our university [10]. These programs saw students learn cybersecurity using physical educational robots as a tool to make the topic more engaging. This work expands on this system both by moving the robots into simulated, networked environments, but also by providing the capability to use the same concepts to teach new domains.

II. NetsBlox

Programming functionality is provided through NetsBlox [11], an extension of the Snap*!* [12] environment with the primary goal of introducing distributed computing features [13]. This allows for students of all skill levels to be provided with block-based abstractions for a focus on computational thinking rather than syntax. While Snap*!* already includes advanced features such as custom block creation and functional programming capabilities, NetsBlox expands upon it with a focus on distributed computing features provided through block-based abstractions.

The first way NetsBlox introduces students to distributed computing is through the ability to pass messages between NetsBlox programs. While projects run entirely on the students' browsers, the server infrastructure for NetsBlox provides addressing for user sessions and projects so that any two running NetsBlox programs can communicate. Students define message types with named fields so that the data sent arrive as variables in the "when I receive msg" hat block. In addition, students are able to share a programming session in real time over the Internet, similar to services such as Replit [14], allowing for collaboratively coded projects.

Fig. 1. Example code accessing web service data through an RPC in NetsBlox

NetsBlox also provides students with a rich set of web services for integration into their projects in the form of Remote Procedure Calls (RPC). RPCs are accessed through a "call" block (if we need the return value) or a "run" block (if we do not). Related RPCs are grouped into *services*. For example, a student making a weather app can connect to an online weather API through the NetsBlox server by using a "call" block to call the remote "temperature" procedure on the "Weather" service, as seen in Figure 1. Services are also provided for features such as cloud storage, location services with maps, access to scientific datasets on topics such as climate change, the COVID-19 pandemic, a movie database and language translation.

A. RoboScape

Robotics support for NetsBlox is provided as a service named RoboScape. [10] This service provides RPCs for giving commands to robots and for receiving sensor values from them. This service was developed for physical robots and it currently supports the Parallax ActivityBot 360 robots.

While typical educational robotics platforms focus on students developing firmware to run on the robot [3], RoboScape continues our goal of making distributed computing concepts more approachable by having students' code instead send commands over the network. For example, Figure 2 shows a simple program that instructs the robot repeatedly to move forward at about half its top speed for one second and then turn in place by setting the speed of the left and right wheels.

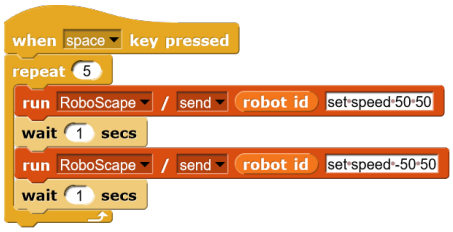

Fig. 2. Example NetsBlox code sending commands to a RoboScape robot

The network is intentionally designed to allow other students to eavesdrop, intercept, and spoof messages to facilitate a cybersecurity curriculum. Students are able to chose an encryption algorithm and set encryption keys for their commands and messages, enable sequence numbers to prevent replay attacks, and more, during a series of challenges where other students are allowed to attempt to disrupt others' robots' operation. Previous educational programs run using RoboScape [10] have demonstrated that this design choice led to fun and engaging cybersecurity education.

III. RoboScape Online

RoboScape Online reimagines the approach for cyberspace. It seeks to provide a more accessible platform for distance computer science education using robotics. Students are provided a shared virtual space to use virtual robots in. At the same time, teachers can utilize tools to help manage the class while they participate in the simulated robotics activities.

The client and server software for RoboScape Online are created in the Unity game engine [15]. A game engine was chosen for development for multiple reasons. Unity is designed to run on multiple operating systems and computer architectures, making it straightforward to provide executable files for Windows, macOS, Linux, and mobile platforms from a single codebase. There is a large ecosystem of both plugins and developer support for Unity, allowing easier integration of

features such as robust networking support. In addition, game engines have the express purpose of running in real-time with an attractive level of graphical fidelity on common consumer-grade hardware.

The use of the Unity game engine also allows for easier content distribution through Unity's AssetBundles. RoboScape Online uses this feature to download remotely hosted scenario files, so a wide array of environments can be made available online without a large upfront download. AssetBundles only allow for models, textures, sounds, and similar assets to be included, rather than custom scripts. Scripts in the client program can be referenced by a scene in an AssetBundle, so a rich library of scripts has been provided to facilitate interactivity in custom environments. While this required additional development time, it comes with the benefit that user-generated content can be trusted more. New content is created through Unity's editor. The amount of support and documentation available online for this editor is significant enough that even students will be able to make impressive scenes.

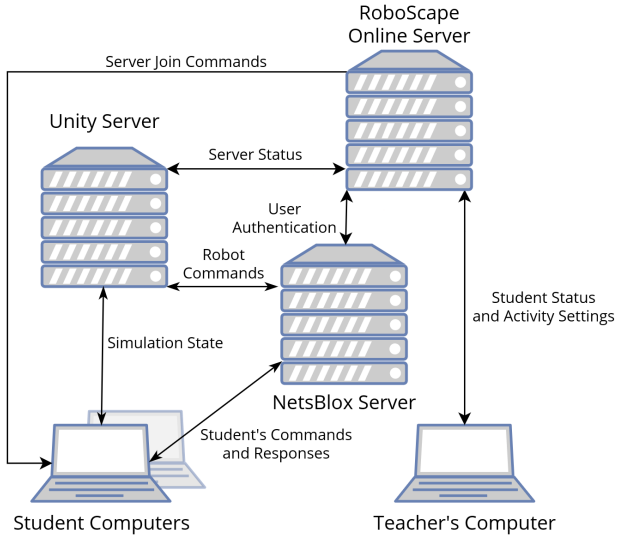

Fig. 3. Architecture of internet connected components

To enable networked robotics simulations, RoboScape Online is distributed across multiple components (See Figure 3). First, there is the main NetsBlox server, which in addition to hosting the programming interface, stores user information for authentication purposes. The second component is the RoboScape Online server the Unity clients and servers connect to. The web interface for teachers to use is designed as an expansion of an existing dashboard provided with NetsBlox. Unity servers running on cloud instances announce themselves and their capacity to the RoboScape Online server. This allows for a flexibility, where new servers can easily be added to the system automatically based on demand. The RoboScape Online server will automatically connect students and servers as needed.

Robots in RoboScape Online default to a model of the Parallax ActivityBot configuration previously used with RoboScape in physical classrooms, but this represents only a basic level of functionality. This robot has two drive motors, front facing "whisker" bumper sensors, an ultrasonic distance sensor, a button, and two LEDs. Simulated robots may be provided with sensors and actuators beyond this, and beyond what would be feasible given the limits of classroom use. GPS, Lidar, inertial measurement units, environmental sensors, and much more can be added to robots based on the needs of the scenario. Simulated sensors will be able to have parameters tweaked to a desired range, or to introduce effects such as bias or noise for added challenge or realism. In addition, it will be possible for a scenario to dynamically change the capabilities of a robot at run-time, both to simulate different conditions and to enforce restrictions for a competition mode. These features make RoboScape Online a truly "low threshold, high ceiling" environment. In addition, the abstractions provided through blocks make student code easily reusable between virtual and physical robots with the same capabilities, thus students working with physical robots may "take home" a virtual robot for homework.

RoboScape Online's web interface will allow teachers to manage their students and control how they interact with the platform. Educators are given the ability to create accounts for students and assign them to groups. When a class is in session, the instructor has access to a dashboard with information on the status of each student. Teachers are given a list of available scenarios to assign to their students, which will automatically connect their class to an appropriate cloud instance running the server program. These scenarios are available as curricula to provide instructors with relevant course materials to distribute to their students, and each curriculum presents multiple options for some scenarios. Robots are automatically assigned an access control profile such that only the relevant students have the ability to interact with them. This access control both controls the ability to send commands to the robot and allows for different groups of students to be able to interact with certain components of a robot. For example, the cybersecurity focused curriculum previously used with physical robots relied on pushing the robot's button to trigger generation of a hardware encryption key, displayed using its LEDs. In a physical classroom, this did not pose any issue, but in a virtual one, it is important that students are restricted from seeing their classmate' keys or pressing the buttons on robots not assigned to their group.

A. *Example Virtual Environments*

The environment seen in Figure 4 is the "box pushing" scenario. Here the students' robots are placed on a platform. Boxes drop from a conveyor belt near the platform, and students are assigned to use the robots to push the boxes off of the platform. While this is a simple task, this scenario demonstrates the multiple ways RoboScape Online can be used. This task is made available as a competitive scenario, where students are given different regions of the ground to push boxes towards, a collaborative scenario, where students work together to remove all boxes, and an individual scenario, where students work alone. Teachers also have the choice for their class to assign this as an autonomous task where the

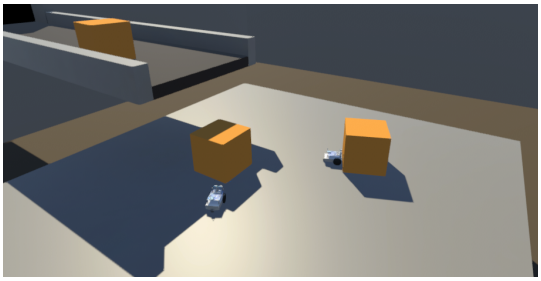

Fig. 4. View of "box pushing" scenario

students' programs must operate based solely on robot sensors, or a manual driving task where students only write a remote control program in NetsBlox that use the arrow keys to issue commands, for example. Scenarios such as this could also be combined with the cybersecurity elements of RoboScape to provide additional challenges for students.

Figure 5 shows the "traffic signal" scenario. This environment demonstrates the platform's ability to extend beyond simulating robots by tasking the students to operate the lights on an internet-connected traffic signal. The signals and a row of vehicle proximity sensors are exposed to the students as web services in NetsBlox. The traffic is generated automatically by the environment. Performance is evaluated by the rate of vehicles passing through the intersection.

Fig. 5. View of "traffic signal" scenario

IV. Evaluation

To evaluate the usefulness of RoboScape Online, a new summer program, similar to the previous ones performed at our university, was conducted this summer. This virtual program allowed student and teacher feedback to be collected. A virtual version of the previous RoboScape curriculum was utilized in the virtual environment with teachers trained to use the system then running a class with it. Most events were performed in a specialized scenario environment, but for some only an empty space with robots was provided. Students worked in groups of two to three, and were required to have completed AP CSP to have some familiarity with programming.

Teachers who participated in the program stated that "Students were engaged and seemed to be having a good time." They did not find the virtual robots to be difficult to work with, and remarked that any technical issues were quickly mediated. Feedback from students included that they enjoyed the collaboration features and the novice-friendly programming environment. Students were able to successfully complete the challenges given to them and generally responded positively. However, some students with previous robotics experience said they preferred working with physical robots.

V. Conclusion

The ongoing demand for expanded distance education can be assisted by a new robotics simulation platform. Robotics education in the classroom provides students with increased engagement, but also presents teachers with multiple costs. Simulating robots eliminates many of these costs while also facilitating distance education. To further remove barriers to entry, RoboScape Online utilizes a block-based programming language with strong distributed computing features. NetsBlox's ability to support student collaboration, augmented by RoboScape Online's networked virtual spaces, allows students to easily work together, and to complete robotics challenges no matter their level of programming expertise or their location. The web interface to RoboScape Online allows teachers to set up their classroom without requiring experience with the more technical aspects of its infrastructure. Instructors are given the option to tweak the scenarios for the curriculum given to them, while also having the potential to create their own novel environments for students to learn in.

The ability to use and create diverse environments will allow this platform to expand beyond simple robot programming tasks. Robotics education has previously been applied to STEM domains such as math [16] [17] and cybersecurity [10], but with freedom to create environments, the platform could be used for simulations of ecology, history, or cultural studies as well. Providing teachers with the tools they need to both create new environments, host shared spaces for their students, and interact with them through a novice-friendly programming interface should create many new opportunities for audiences and ideas which were previously underrepresented in cyber-physical systems education.

A. Future Work

By providing an open platform for virtual robotics, RoboScape Online will assist other researchers to conduct studies on simulated robotics education. To continue in this direction, additional research tools must be added. The software currently has limited logging support for student commands and robot positions, but this can be extended with additional analytics features.

A set of VR tools is under development to create a "virtual makerspace." This would expand the concept to also support the creation of novel robot designs by the students themselves, adding potential for richer engineering challenges. It is possible that the increased immersion provided by a head-mounted display will provide an experience more engaging to students with existing robotics experience.

With the feature set of the Unity engine allowing for audio/video content to be included in scenario files, the potential exists to implement entirely virtual lessons where instruction is included in the environment itself. Students could be given more freedom to select topics and lessons could provide smart assistance similar to what iSnap [18] does for Snap*!*.

ACKNOWLEDGMENT

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. 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

[1] O. B. Adedoyin and E. Soykan, "Covid-19 pandemic and online learning: the challenges and opportunities," *Interactive Learning Environments*, pp. 1–13, 2020.

[2] S. Anwar, N. A. Bascou, M. Menekse, and A. Kardgar, "A Systematic Review of Studies on Educational Robotics," *Journal of Pre-College Engineering Education Research (J-PEER)*, vol. 9, no. 2, 2019.

[3] L. Xia and B. Zhong, "A systematic review on teaching and learning robotics content knowledge in k-12," *Computers & Education*, vol. 127, p. 267–282, 2018.

[4] S. Tselegkaridis and T. Sapounidis, "Simulators in Educational Robotics: A Review," *Education Sciences*, vol. 11, no. 1, p. 11, 2021.

[5] E. B. Witherspoon, R. M. Higashi, C. D. Schunn, E. C. Baehr, and R. Shoop, "Developing Computational Thinking through a Virtual Robotics Programming Curriculum," *ACM Transactions on Computing Education (TOCE)*, vol. 18, no. 1, p. 4, 2017.

[6] O. S. R. Foundation, "Gazebo website," 2021. [Online]. Available: http://gazebosim.org/

[7] "Robotify website," https://robotify.com, 2021, cited 2021 March 8.

[8] Y. Wang, Y. Chen, X. Tong, Y. Lee, and J. Yang, "Robot as a service in information science & electronic engineering education," *2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS)*, p. 223–228, 2017.

[9] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The Robotarium: A Remotely Accessible Swarm Robotics Research Testbed," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1699–1706, 2017.

[10] A. Lédeczi, M. Maroti *et al.*, "Teaching cybersecurity with networked robots," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, Feb 2019, p. 885–891. [Online]. Available: 10.1145/3287324.3287450

[11] "Netsblox website," https://netsblox.org, 2021, cited 2021 March 8.

[12] "Snap! website," https://snap.berkeley.edu/, 2021, cited 2021 March 8.

[13] B. Broll, M. Lu, A. Ledeczi, and et al., "A visual programming environment for learning distributed programming," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, Mar 2017, pp. 81–86. [Online]. Available: 10.1145/3017680.3017741

[14] "Replit website," https://replit.com, 2021, cited 2021 March 8.

[15] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A General Platform for Intelligent Agents," *arXiv*, 2018.

[16] B. Zhong and L. Xia, "A Systematic Review on Exploring the Potential of Educational Robotics in Mathematics Education," *International Journal of Science and Mathematics Education*, vol. 18, no. 1, pp. 79–101, 2020.

[17] C. Chung and E. Santos, "Robofest Carnival — STEM Learning Through Robotics with Parents," *2018 IEEE Integrated STEM Education Conference (ISEC)*, pp. 8–13, 2018.

[18] T. W. Price, Y. Dong, and D. Lipovac, "iSnap: Towards Intelligent Tutoring in Novice Programming Environments," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18*, ser. Proceedings of the 49th ACM Technical Symposium on Computer Science Education, 3 2017, pp. 1113–1113.