Position: INTELLIBLOX: A Toolkit for Integrating Block-Based Programming into Game-Based Learning Environments

Sandra Taylor
Dept. of Computer Science
North Carolina State University
Raleigh, NC, USA
smtayl23@ncsu.edu

Wookhee Min
Dept. of Computer Science
North Carolina State University
Raleigh, NC, USA
wmin@ncsu.edu

Bradford Mott
Dept. of Computer Science
North Carolina State University
Raleigh, NC, USA
bwmott@ncsu.edu

Andrew Emerson
Dept. of Computer Science
North Carolina State University
Raleigh, NC, USA
ajemerso@ncsu.edu

Andy Smith
Dept. of Computer Science
North Carolina State University
Raleigh, NC, USA
pmsmith4@ncsu.edu

Eric Wiebe
Dept. of STEM Education
North Carolina State University
Raleigh, NC, USA
wiebe@ncsu.edu

James Lester
Dept. of Computer Science
North Carolina State University
Raleigh, NC, USA
lester@ncsu.edu

***Abstract*—Block-based programming languages reduce the need to learn low-level programming syntax while enabling novice learners to focus on computational thinking skills. Game-based learning environments have been shown to create effective and engaging learning experiences for students in a broad range of educational domains. The fusion of block-based programming with game-based learning offers significant potential to motivate learners to develop computational thinking skills. A key challenge educational game developers face in creating rich, interactive learning experiences that integrate computational thinking activities is the lack of an embeddable block-based programming toolkit. Current block-based programming languages, such as Blockly and Scratch, cannot be easily embedded into industry-standard 3D game engines. This paper presents INTELLIBLOX, a Blockly-inspired toolkit for the Unity cross-platform game engine that enables learners to create block-based programs within immersive game-based learning environments. Our experience using INTELLIBLOX suggests that it is an effective toolkit for integrating block-based programming challenges into game-based learning environments.**

Keywords—block-based programming, game-based learning, K-12 computer science education, computational thinking

I. INTRODUCTION

Block-based programming has proven effective in teaching computational thinking (CT) skills to novice learners by decreasing cognitive load, promoting recognition of shapes over recall of text-based syntax, and limiting the ability to create syntax errors [1]. Block-based programming environments have been used to create engaging learning experiences to introduce students to programming, including environments that support the creation of games and interactive artifacts [2], [3], [4], [5]. Likewise, game-based learning environments have been used to provide students with rich, immersive learning experiences that have been carefully crafted to promote student motivation and learning outcomes [6]. Consequently, the integration of block-based programming into the core gameplay of game-based learning environments offers significant promise for supporting K-12 students in developing computational thinking skills [7]. A key challenge facing educational game developers seeking to leverage block-based programming in game designs is the lack of a framework for embedding block-based programming into immersive 3D games. While there are several open source block-based programming toolkits (Blockly, Snap!, Scratch), incorporating them into 3D game engines is labor-intensive and limits their ability to be tightly integrated with gameplay. This paper presents a toolkit for integrating block-based programming within games created with the Unity cross-platform game engine.

II. INTELLIBLOX

INTELLIBLOX is a toolkit that enables block-based programming to be tightly integrated into the core gameplay of immersive game-based learning environments developed using the Unity game engine. Unity is a cross-platform game development environment for creating 2D and 3D games on computers, video game consoles, and mobile devices used by millions of game developers worldwide. It is a popular platform for building game-based learning environments for K-12 education because it supports tablet and WebGL builds of games that run on iOS and Android tablets as well as Chromebooks, which are quickly becoming the platform of choice in schools. Thus, INTELLIBLOX facilitates bringing the significant benefits of block-based programming to a wide range of engaging gameplay activities. The design of INTELLIBLOX takes into account key CT practices—developing and using abstractions, creating computational artifacts, and testing and refining computational artifacts—suggested by the K-12 Computer Science Framework [8] in order to effectively support the development of CT skills for students immersed in game-based learning. In addition, INTELLIBLOX is designed to support adaptive scaffolding in game-based learning by logging

978-1-7281-4849-6/19/$31.00 ©2019 IEEE

fine-grained trace data linked to key computational thinking practices and skills.

A. Requirements

Building on our prior experience developing game-based learning environments [9], [10] and integrating block-based programming within a game-based learning environment for middle school computational thinking [7], we have identified a set of core requirements for integrating block-based programming into immersive game-based learning environments. These include the following key requirements.

Integration with an industry-standard game engine. First and foremost, INTELLIBLOX should provide a seamless integration into a full-featured, cross-platform game engine. This integration should allow for the direct manipulation of 2D and 3D game objects, providing the ability to create fully immersive experiences leveraging block-based programming that are not easily created using simpler web-based toolkits. The game engine should also be optimized for deploying WebGL, Windows, macOS, Android, and iOS versions of the game-based learning environment to support widespread K-12 classroom implementations.

Enable adaptive scaffolding. Adaptive scaffolding, which includes providing students with just-in-time feedback and task selection, is critical to supporting effective learning in the classroom [11]. To provide adaptive scaffolding, it is crucial for INTELLIBLOX to enable analysis of student interactions during block-based programming and to support dynamic inference of students' knowledge and skills. Thus, INTELLIBLOX should log and report key features from student interactions at a fine-grained level. In game-based learning environments, student programming trace data can be used to assess student knowledge and skills in the context of stealth assessment [7] and extract salient patterns in students' learning behaviors [12]. By logging student data at a fine-grained level, it is possible to analyze student behaviors to proactively provide feedback and hints to students [13]. Additionally, by supporting methods that can analyze the sequential structure of the logged events, further analysis can capture points in a particular student's interaction where he or she is struggling and failing to make further progress.

Promote computational thinking. A key requirement for INTELLIBLOX is to foster computational thinking (CT) in K-12 students through a rich programming environment. CT involves several key practices, including using abstractions, algorithmic thinking, systematic information processing, and leveraging computational tools for data analysis, modeling, or simulation [14]. To support this, a rich set of programming blocks should be provided, including variables, loops, conditionals, modularity, and event handlers, as well as domain-specific blocks, which can be collectively utilized to construct programs. In association with the adaptive scaffolding requirement, the programming blocks available to students should be dynamically chosen and presented to students according to students' skills and knowledge.

Flexible block palette. To facilitate instructional scaffolding in block-based programming environments, it is helpful to limit the number of blocks presented to beginning students [15] and gradually increase the number of blocks as students' skills improve to prevent early frustration. In addition, the ability to organize blocks into functionally similar palettes is helpful.

Rich developer toolset. To ensure scalability, expedite the addition of new programming blocks, and facilitate independent testing outside of the game environment, it is important to provide a rich developer toolset as well as developer documentation [16] for creating and testing custom blocks and building block palettes.

De facto standard block-based programming. To create an effective block-based programming environment, it is important to use broadly accepted standards for block-based programming languages within INTELLIBLOX [17]. More specifically, INTELLIBLOX should provide a drag and drop-based workspace containing interconnectable blocks displaying natural language which provide graphical cues that reinforce the rules of composition in order to reduce errors and frustration for beginning learners.

B. Design and Architecture

We highlight the key design and architectural decisions in the implementation of INTELLIBLOX below.

Game engine selection. With a focus on developing immersive 3D learning environments targeting multiple platforms, we selected the Unity cross-platform game engine as the basis for INTELLIBLOX. Unity allows for rendering the block-based programming UI on 2D UI objects as well as 3D objects within the game world. Unity is optimized for several platforms commonly used in K-12 classrooms, including WebGL, Windows, macOS, Android, and iOS.

Leveraging Blockly. Blockly is an open source JavaScript library for building visual programming editors [18] that is best suited for integration within web-based applications as opposed to 3D game engines. Though not ideal for use in Unity-based learning environments, Blockly does provide a feature-rich and well-established framework as an excellent starting point for developing INTELLIBLOX. As such, we have leveraged as much of Blockly's existing open source code as possible. Specifically, we have used Blockly's grammar, Blockly's data model, the Blockly developer toolset, and Blockly's "blocks-to-code" generation mechanism. By using these features and capabilities, INTELLIBLOX supports an easily extendable block palette and the rapid creation of additional blocks necessary for incorporating K-12 computational thinking concepts. In addition, we have based the look and feel of our blocks on Blockly.

While Blockly provides a good foundation, it should be noted that there are several key missing components that are required for creating an immersive game-based learning environment featuring block-based programming gameplay.

Direct manipulation of game objects by block-based programs. To enable students' block-based programs to manipulate the gameworld, we have created a code generation server based on Blockly and Node-blockly [19], which is accessed via a REST API to translate block-based programs into executable Lua code. This Lua code is then interpreted and executed using MoonSharp [20], an open source Lua interpreter that supports the execution of Lua code in Unity. This enables access to the full Unity API through this block-based program to Lua to Unity bridge.

Block-based programming editor in Unity. INTELLIBLOX features a drag and drop block-based program editor implemented using the Unity UI system. INTELLIBLOX blocks are dynamically generated at run-time as a series of Unity UI components based on JSON definitions that are consistent with Blockly block definitions.

Interaction logs. To support adaptive scaffolding, INTELLIBLOX supports the logging of all UI interactions, such as creating and deleting blocks, connecting and disconnecting blocks, changing block parameters, and current block locations in 2D space as well as a progressive sequence of programming artifacts. This event data stream supports the recording and playback of coding sessions to facilitate analysis of students' coding behavior to support real-time scaffolding [21].

III. ENGAGE: A GAME-BASED LEARNING ENVIRONMENT BUILDING ON INTELLIBLOX

ENGAGE (Figure 1) is an immersive game-based learning environment designed to develop computational thinking skills for middle school students [7]. ENGAGE uses INTELLIBLOX to integrate computational challenges into a rich storyworld. In ENGAGE, students play the role of the protagonist who has been sent to an undersea research facility to determine why all communication with the station has been lost. As students explore the station, they must progress through multiple levels consisting of a series of rooms. Each room presents students with a set of computational challenges they must overcome using the INTELLIBLOX editor in order to advance to the next area. For example, students are prompted to operate a quadcopter (Figure 1) by writing a program using the INTELLIBLOX editor (Figure 2) in order to get across a pit. Students have to use multiple *rotate* blocks along with multiple *move forward* blocks, or more efficiently with *repeat* blocks to complete the task. As students progress in the game, they encounter more complex computational challenges such as writing a bubble sort algorithm and a data analysis activity. All of the challenges within ENGAGE are in service of the game's narrative and are directly associated with concepts and practices defined in the K-12 Computer Science Framework [8]. For example, the framework's Cybersecurity concept is conveyed in ENGAGE challenges where students are required to write a program to crack a security code via a brute force method.

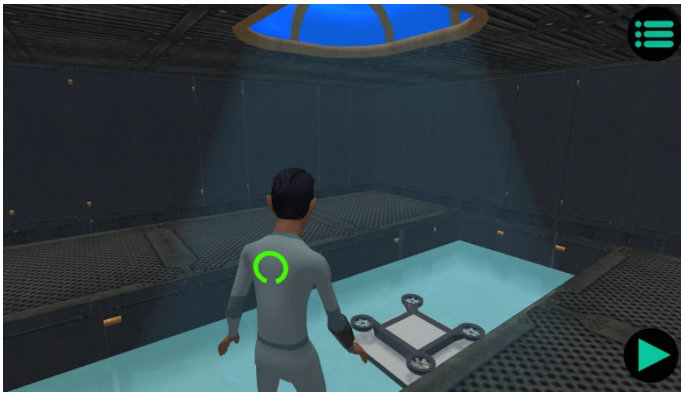

Fig. 1. Screenshot of the ENGAGE game-based learning environment

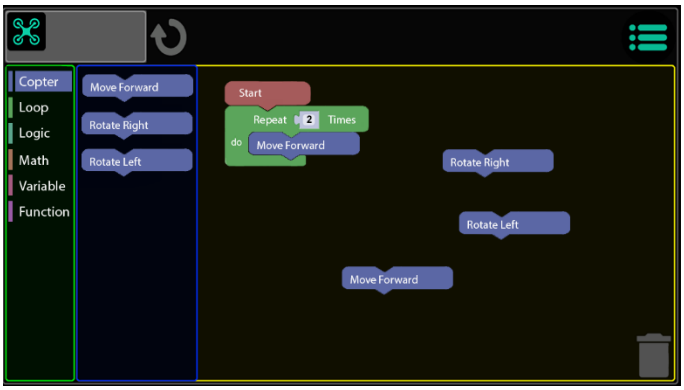

Fig. 2. Screenshot of the INTELLIBLOX editor integrated into ENGAGE

IV. CONCLUSION

Fusing block-based programming with game-based learning offers significant potential for motivating learners to develop computational thinking skills. We have presented INTELLIBLOX, a block-based programming toolkit that integrates with an industry standard game engine, which enables educational game developers to create immersive game-based learning environments featuring block-based programming gameplay challenges.

Initial results using INTELLIBLOX to develop ENGAGE, an immersive game-based learning for middle school students, suggest that INTELLIBLOX is an effective toolkit. We are also leveraging INTELLIBLOX to create a new game-based learning environment that engages upper elementary students in immersive AI learning experiences as well as a digital storytelling environment for upper elementary students that features problem-based learning scenarios integrating computational thinking with physical science activities. As INTELLIBLOX matures through the design and development of these learning environments, we will release it as an open source project that can be used to develop other immersive game-based learning environments.

ACKNOWLEDGMENT

The authors wish to thank colleagues from the IntelliMedia Group for their assistance with this work. This research was supported by the National Science Foundation under Grants DRL-1640141, DRL-1934153, and DRL-1921495. 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 National Science Foundation.

REFERENCES

[1] D. Bau, J. Gray, C. Kelleher, J. Sheldon, and F. Turbak, "Learnable programming: Blocks and beyond," *Commun. ACM.* vol. 60, pp. 72–80, 2017.

[2] "Gameblox," https://gameblox.org, 2019.

[3] "Microsoft Makecode for Minecraft," https://minecraft.makecode.com. 2019.

[4] "Scratch for Second Life," http://web.mit.edu/~eric_r/Public/S4SL/, 2019.

[5] "MIT Media Lab Lifelong Kindergarten: Project Microworlds," https:// www.media.mit.edu/projects/microworlds/overview/, 2019.

[6] M. Qian and K. R. Clark, "Game-based learning and 21st century skills: A review of recent research," *Computers in Human Behavior*, vol. 63, pp. 50–58, 2016.

[7] W. Min, M. Frankosky, B. Mott, J. Rowe, A. Smith, E. Wiebe, K. E. Boyer, and J. Lester. "DeepStealth: game-based learning stealth assessment with deep neural networks," *IEEE Transactions on Learning Technologies*, in press.

[8] "K–12 computer science framework," https://k12cs.org/, 2018.

[9] J. P. Rowe, L. R. Shores, B. W. Mott, and J. C. Lester, "Integrating learning, problem solving, and engagement in narrative-centered learning environments," *International Journal of Artificial Intelligence in Education*, vol. 21, no. 1–2, pp. 115–133, 2011.

[10] J. C. Lester, E. Y. Ha, S. Lee, B. W. Mott, J. P. Rowe, and J. Sabourin, "Serious games get smart: Intelligent game-based learning environments," *AI Magazine*, vol. 34, no. 4, pp. 31–45, 2013.

[11] K. VanLehn, "The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems," *Educational Psychologist*, vol. 46, no. 4, pp. 197–221, 2011.

[12] T. W. Price and T. Barnes. "Comparing textual and block interfaces in a novice programming environment," In *Proc. of the Eleventh Annual International Conference on International Computing Education Research (ICER '15)*, pp. 91–99, 2015.

[13] T. W. Price, R. Zhi, and T. Barnes. "Hint generation under uncertainty: the effect of hint quality on help-seeking behavior," in *Proc. of the International Conference on Artificial Intelligence in Education (AIED'17)*, pp. 311–322, 2017.

[14] S. Grover and R. Pea, "Computational thinking in K-12: a review of the state of the field," *Educational Researcher*, vol. 42, no. 1, pp. 38–43, 2013.

[15] M. Tsur, and N. Rusk. "Scratch microworlds: Designing project-based introductions to coding," in *Proc. of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*, pp. 894-899, 2018.

[16] S. Dasgupta, S. M. Clements, A.Y. Idlbi, C. Willis-Ford, and M. Resnick. "Extending scratch: New pathways into programming," in Proc. of 2015 *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC '15)*, pp. 165-169, Oct. 2015.

[17] D. Weintrop and U. Wilensky, "To block or not to block, that is the question: Students' perceptions of blocks-based programming," in *Proc. of the 14th International Conference on Interaction Design and Children*, pp. 199–208, 2015

[18] N. Fraser, "Ten things we've learned from blockly," in *Proc. of 2015 IEEE Blocks and Beyond Workshop*, pp. 49–50, 2015.

[19] "Node-blockly," https://github.com/mo4islona/node-blockly, 2019.

[20] "MoonSharp," https://www.moonsharp.org/, 2019.

[21] A. Emerson, F. Rodríguez, B. Mott, A. Smith, W. Min, K. Boyer, C. Smith, E. Wiebe, and J. Lester, "Predicting early and often: Predictive student modeling for block-based programming environments," *12th International Conference on Educational Data Mining*, pp. 39–48, 2019.