Kong, S. C., Sheldon, J., & Li, K. Y.. (Eds.). (2017). *Conference Proceedings of International Conference on Computational Thinking Education 2017*. Hong Kong: The Education University of Hong Kong

App Inventor VR Editor for Computational Thinking

Jane IM [1], Paul MEDLOCK-WALTON[2*], Mike TISSENBAUM[2*]

[1] Korea University
[2] Massachusetts Institute of Technology
jane605@korea.ac.kr, paulmw@mit.edu, mtissen@mit.edu

ABSTRACT

This paper introduces the concept of a virtual reality (VR) programming environment that allows youth to both develop immersive VR experiences while enhancing computational thinking (CT). Specifically, we extended a blocks-based programming platform, MIT App Inventor, to allow youth to make VR Android apps (AI/VR). We compare AI/VR's support for CT to other existing VR editors using the CT concepts established by Brennan and Resnick (2012). Comparisons showed that AI/VR's support for all CT concepts and its ease of use for kids, makes it more preferable for teaching CT compared to other editors.

KEYWORDS

computational thinking, virtual reality, constructionism, immersive interface, MIT App Inventor

1. INTRODUCTION

In recent years, many educators have argued computational thinking (CT) (Wing, 2006) is an indispensable skill for everyone. In order to support widespread uptake of computational thinking, blocks-based approaches to programming have been developed, in which users program by snapping blocks of code. For example, Scratch allows students to build 2D multimedia (Brennan & Resnick, 2012). Alice helps students learn programming by building 3D media (Dann, Cooper, & Pausch, 2006).

Compared to Scratch, Alice provided a more immersive experience. Studies using Alice showed it is effective for learning programming, in part to its immersive nature (Sykes, 2007), indicating the potential for immersive experiences to enhance students' computational thinking. However, there has been limited research on how VR, an immersive environment, can support CT learning. Given the nascent field of VR, in order to understand the role of it in developing CT, there is a need to examine current VR editors. If these do not support the kinds of learning we wish to support, then it is critical to develop appropriate tools. This work was framed around two needs: 1) understand the state of current VR editors and examine their suitability for supporting developing computational thinking; and 2) develop a tool that supports the learning needed, if current platforms were found to be lacking.

Below, we examine current VR editors, discuss how they support CT and their suitability for young learners. We propose AI/VR that responds to their shortcomings.

2. BACKGROUND

2.1. *Constructionism*

Constructionism is the process of building understanding through the active use of tools to develop tangible artifacts (Kafai & Resnick, 2011). Building on constructionism, is the concept of "learning as designers", which has shown to increase higher-order thought process development and motivation (Cooper, Dann, & Pausch 2003; Fortus, Dershimer, Krajcik, Marx, & Mamlok-Naaman, 2004). Especially, programming interactive media has been shown to support CT (Brennan & Resnick, 2012). The development of interactive and immersive media, with platforms such as Alice, also embody constructionist characteristics, as they allow learners to design interactive media freely in the same context (Sykes, 2007).

2.2. *Immersive Interface for Learning*

Immersion is the subjective impression that one is participating in a comprehensive, realistic experience (Stanney, 2002; Lessiter, Freeman, Keogh, & Davidoff, 2001). Studies have shown that immersion in a digital environment can enhance education in at least three ways: allowing multiple perspectives, situating the learning, and transfer to other contexts (Dede, 2009). Below we describe two immersive learning environments.

2.3. *Alice*

Alice is a 3D graphics programming environment that allows users to create interactive 3D animations and learn programming in an object-oriented approach (Dann et al., 2006). Research on use of Alice to teach an entry level undergraduate computer science course showed that posttest performance among students who used Alice was significantly higher than comparison groups. Qualitative results showed that students using Alice enjoyed the process and spent more time engaged in the course (Sykes, 2007). There were diverse reasons for this engagement, including

the active graphical interface. While Alice is not completely immersive, it has a higher degree of immersion compared to text-based languages. The results suggest the potential for enhancing students' computational thinking skills within a more immersive environment.

2.4. *Immersive Interface for Learning*

Virtual Environment Interactions (VEnvI) is a platform that uses a database of dance sequences, VR, and a drag-and-drop interface to teach programming concepts

(Parmar et al., 2016). Although this study is limited because students did little programming, and were introduced to programming concepts in sessions, results showed that students found the immersion of VEnvI desirable and became more positive towards computer science.

3. PREVIOUS EDITORS

3.1. *Introduction of VR Editors*

In order to understand how current platforms support CT, we examined nine VR code editors: 360°& VR Editor ("360°& VR Editor", n.d), HoloBuilder ("HoloBuilder", n.d.), Smart2VR ("Smart2VR", n.d.), CoSpaces ("CoSpaces", n.d.), Vizor ("Vizor", n.d.), Unity ("VR Overview", n.d.), Unreal Engine VR Editor ("Unreal Engine VR Editor", n.d.), Arma 3's virtual reality editor (Arma 3 has a VR editor for creating games) (Zemánek, 2014), and Simmetri ("Simmetri", n.d.).

3.2. *Categorization and Analysis*

We categorized the editors using Brennan and Resnick's computational concepts, which are sequences, loops, parallelism, events, conditionals, operators, and data (Brennan & Resnick, 2012). The editors are also categorized based on their affordances into three groups, which are photo/video focused editors, visual programming editors, and text based editors (*Table 1*).

3.2.1. *Photo or video focused editors*

Photo or video focused editors are ones that: 1) focus on making rich scenes using photos and videos; and 2) only support acquiring the computational thinking concept 'events'. The focus on scene creation is the goal of these editors, which explains why they have limited utility for CT. Users can place events inside scenes using drag-and-drop (e.g., adding a button that is clickable). However, such editors lack the means to use sequences, loops, parallelism, events, operators, and data. 360°& VR Editor, HoloBuilder, and Smart2VR fit in this category.

3.2.2. *Visual programming editors*

Visual programming editors support most, if not all, the CT concepts through visual programming. Vizor and CoSpaces fall in this category, with CoSpaces also supporting text based programing.

In Vizor, students can apply all computational thinking concepts through using a 'patch', which can be connected to other patches (*Figure 1*). Users can combine patches like state/structure patches to understand sequence. There are prebuilt patches for loops, conditionals, operators, variables, and data. Users can learn parallelism, for example, by using two mouse press patches. However, there are limitations including the limited animation patches, which can make animating objects difficult. Additionally, unlike MIT App Inventor where blocks run from top to bottom, the order of patches do not indicate sequence in Vizor, requiring users to link extra patches.

In CoSpaces, users can use blocks to apply all seven CT concepts. Users can connect blocks from top to bottom to understand sequence, and use loop blocks to understand loops. The *execute in parallel* and the *on activate of* blocks enable parallelism and events in users' projects, respectively. There are prebuilt blocks for conditionals, operators, variables and data. However, CoSpaces lacks blocks for dynamically creating objects, and has limited types of events compared to Vizor and AI/VR. These can limit the range of computational practices (which focus on "how", instead of "what" users learn) (Brennan & Resnick, 2012).

Table 1. Categorization of VR Editors with colored boxes representing an attribute(column) that an editor(row) has. Numbers 1,2,3,4,5,6,7 of CT concept each refer to sequences, loops, parallelism, events, conditionals, operators, and data.

		CT Concepts								
Platform	Editor Type	1	2	3	4	5	6	7	Does not require programming background	Intended audience
360°& VR Editor	Photo/Video				■				■	Novices
HoloBuilder					■				■	Novices, especially construction company
Smart2VR					■				■	Novices
Vizor	Visual	■	■	■	■	■	■	■	■	Novices
CoSpaces	Visual/Text based	■	■	■	■	■	■	■	■	Novices
Unity	Text based	■	■	■	■	■	■	■		Professional, Experienced gamers
Unreal Engine VR Editor		■	■	■	■	■	■	■		Professional, Experienced gamers
Arma 3's editor		■	■	■	■	■	■	■		Professional, Experienced gamers
Simmetri		■	■	■	■	■	■	■	■	Artists
AI/VR	Visual	■	■	■	■	■	■	■	■	Novices

Figure 1. Usage of patches in Vizor.

3.3. *Text-based editors*

Text based editors are editors that require at least partial text based programming to exhibit the seven CT concepts. Unity, Unreal Engine VR Editor, Arma 3's virtual reality editor, and Simmetri fit in this category, with differing audiences (*Table 1*). Since these editors allow users to build complicated VR environments technically, they support all CT concepts, and users can employ complex CT with them. However, these text-based editors require a steep learning curve and are not suitable for beginners.

Building off of the various shortcomings of the tools described above, we identified a gap in the VR authoring landscape for a tool that allows novices to develop VR applications while developing CT concept understandings. Below we describe the tool and its use.

4. APP INVENTOR VR EDITOR

4.1. *Blocks in AI/VR*

There are four kinds of blocks in AI/VR: 1) **event**, 2) **method**, 3) **property setter and getter**, and 4) **object creation** blocks. **Event** includes *checkButton*, which checks if the user clicks the Cardboard button. **Method** blocks trigger interactions with objects and the player, including *moveUser*, which changes the location of the player. **Property setters and getters** change object's attributes, such as size. **Object creation** blocks, such as *createCube*, allow the user to dynamically add objects.

4.2. *Sample AI/VR program*

To demonstrate how the editor supports CT, we included a sample AI/VR program (*Figure 2* & *Figure 3*). If the user gazes at one of the four cubes (that are made by shaking the phone), she will earn points (shown in a label). The cube also moves to a random position and changes color.

This example shows how all seven computational concepts are supported in AI/VR. First, users can understand sequences by checking that blocks are executed in order when the user gazes at a cube. The cube changing color, and the score increasing and updating shows sequentially (① of *Figure 3*). Loops are used to iterate over cubes in ②. Events are used through blocks like *checkGazeShort* (the block that returns 1 if the user gazed at the object - ③). For parallelism, two "*if*" blocks are used to check whether a cube was gazed at and the color and position of the cube are changed concurrently (④). For conditionals, the user can connect event blocks and attribute or animation related blocks with an "*if*" block (⑤). For operators, users can practice addition by adding 1 to the current score when a cube is gazed at (⑥-1) and checking that the increased score is updated in the scene (⑥-2). Lastly, users can understand data by keeping track of cubes using a list block (⑦).

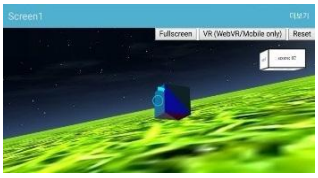

Figure 2. Scene of demo

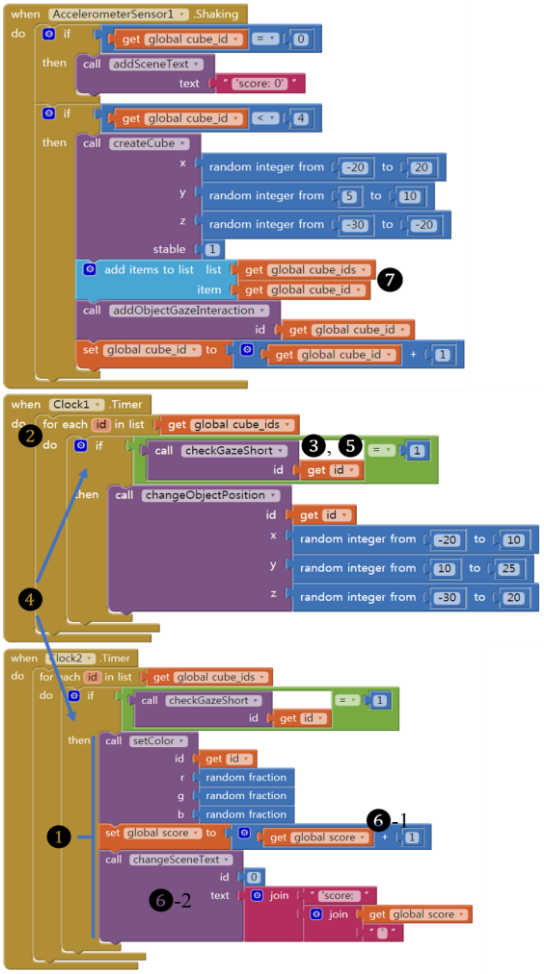

Figure 3. Code for a sample VR program in AI/VR.

4.3. *Categorization and Analysis*

AI/VR is in the category of visual programming editors and supports all CT concepts. AI/VR also targets ease of use by kids, employing the same drag-and-drop interface as the original MIT App Inventor (Wolber, Abelson, Spertus, & Looney, 2011). Considering these aspects, AI/VR balances usability for kids and features for supporting CT, overcoming the limitations of other editors. It also has animation blocks such as *moveObject*, overcoming the limitation of Vizor. Using the top down approach, sequences are also easier in AI/VR. Compared to CoSpaces, AI/VR supports creating new objects with blocks like *createCube*, and allows diverse triggering events with blocks like *checkButton*, which triggers an event when a user presses the Cardboard headset button.

However, AI/VR lacks a diversity of objects and media related blocks like video. In context of CT, this could be

a limitation because it could reduce the diversity of computational practices.

5. CONCLUSION

Considering the role constructionism plays in computational thinking and the possibility of immersive virtual reality to support this learning, we introduce AI/VR - a blocks-based tool to support kids to more easily create virtual reality apps. We have shown AI/VR's fit as a visual programming editor that supports all seven of Brennan and Resnik's CT concepts, while also being usable by young kids. Although AI/VR has limited diversity in objects and media related blocks, it overcomes the limitations of Vizor, such as animating objects and sequence, and those of CoSpaces such as the lack of blocks for dynamically creating objects and the limited types of triggers for events.

6. ACKNOWLEDGEMENT

We would like to thank Hal Abelson, Professor of EECS at MIT, whose insight was great help to this research.

7. REFERENCES

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association (AERA 2012)*.

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. *Proceedings of the 34th SIGCSE technical symposium on Computer science education - SIGCSE '03*.

CoSpaces. (n.d.). Retrieved from https://cospaces.io/create.html

Dann, W., Cooper, S., & Pausch, R. (2006). *Learning to program with Alice.* Upper Saddle River, NJ: Pearson Prentice Hall.

Dede, C. (2009). Immersive interfaces for engagement and learning. *Science, 323*(5910), 66-69.

Fortus, D., Dershimer, R. C., Krajcik, J., Marx, R. W., & Mamlok-Naaman, R. (2004). Design-based science and student learning. *Journal of Research in Science Teaching, 41*(10), 1081-1110.

HoloBuilder (n.d.). Retrieved from http://landing.holobuilder.com/construction

Kafai, Y. B., & Resnick, M. (2011). *Constructionism in practice: Designing, thinking, and learning in a digital world*. New York, NY: Routledge.

Lessiter, J., Freeman, J., Keogh, E., & Davidoff, J. (2001). A cross-media presence questionnaire: The ITC-Sense of Presence Inventory. *Presence: Teleoperators and Virtual Environments, 10*(3), 282-297.

Parmar, D., Isaac, J., Babu, S. V., D'souza, N., Leonard, A. E., Jorg, S., . . . Daily, S. B. (2016). Programming moves: Design and evaluation of applying embodied interaction in virtual environments to enhance computational thinking in middle school students. *2016 IEEE Virtual Reality (VR)*, 131-140.

Simmetri. (n.d.). Retrieved from http://simmetri.com/

Smart2VR. (n.d.). Retrieved from https://www.smart2vr.com/#how-it-works

Stanney, K. M. (2002). *Handbook of virtual environments: design, implementation, and applications.* Mahwah, NJ: Lawrence Erlbaum Associates.

Sykes, E. (2007). Determining the effectiveness of the 3D Alice programming environment at the computer science I level. *Journal of Educational Computing Research, 36*(2), 223-244.

360°& VR Editor. (n.d.). Retrieved from http://demo.thinglink.com/vr-editor

Unreal Engine VR Editor. (n.d.). Retrieved from https://docs.unrealengine.com/latest/INT/Engine/Editor/VR/

Vizor. (n.d.). Retrieved from http://vizor.io/about

VR Overview. (n.d.). Retrieved from https://unity3d.com/kr/learn/tutorials/topics/virtual-reality/vr-overview

Wang, T., Mei, W., Lin, S., Chiu, S., & Lin, J. M. (2009). Teaching programming concepts to high school students with Alice. *2009 39th IEEE Frontiers in Education Conference*, 1-6.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wolber, D., Abelson, H., Spertus, E., & Looney, L. (2011). *App Inventor – Create Your Own Android Apps.* Sebastopol, CA: O'Reilly.

Zemánek, J. (2014, July 15). Arma3: Virtual Reality Custom Courses. Retrieved January 10, 2017, from https://community.bistudio.com/wiki/Arma3:_Virtual_Reality_Custom_Courses