Cubely: Virtual Reality Block-based Programming Environment

Juraj Vincur
Slovak University of Technology
Bratislava, Slovakia
juraj.vincur@stuba.sk

Martin Konopka
Slovak University of Technology
Bratislava, Slovakia
martin_konopka@stuba.sk

Jozef Tvarozek
Slovak University of Technology
Bratislava, Slovakia
jozef.tvarozek@stuba.sk

Martin Hoang
Slovak University of Technology
Bratislava, Slovakia
xhoang@stuba.sk

Pavol Navrat
Slovak University of Technology
Bratislava, Slovakia
pavol.navrat@stuba.sk

ABSTRACT

Block-based programming languages are successfully being used as an alternative way of teaching introductory programming concepts. The success is in part due to the low barrier of entry and the visual game-like appeal fostering experimentation and creativity. Virtual reality (VR) presents a step further to even more immersive experience. In this project, we designed an immersive VR programming environment in which novice programmers solve programming puzzles within a virtual world. The puzzles are similar to Code.org exercises, and the solution (program itself) is assembled by the programmer within the same virtual world. A program in this VR environment consists of cubes (blocks) assembled into program structures, while the program execution is traced to individual cubes and can be directly observed in the virtual world. We evaluated usability of the VR programming environment and students' perceptions in comparison to the usual 2D block-based programming interface. Students found it easy to work with the new VR interface, and slightly preferred using the VR interface over the traditional 2D block-based programming.

CCS CONCEPTS

•**Applied computing** → **Interactive learning environments;**
•**Human-centered computing** → *Virtual reality;*

KEYWORDS

virtual reality, virtual learning environment, visual programming

ACM Reference format:
Juraj Vincur, Martin Konopka, Jozef Tvarozek, Martin Hoang, and Pavol Navrat. 2017. Cubely: Virtual Reality Block-based Programming Environment. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology, El Gothenburg, Sweden, November 2017 (VRST'2017),* 4 pages.
DOI: 10.475/123_4

1 INTRODUCTION

Introducing novices to programming has been of many educators' and researchers' interest, with the main objective being the same – to teach programming in engaging and intuitive way. Proposed approaches mostly abstract from actual low-level programming languages to domain-specific languages designed for specific tasks, e.g., to move a character (robot, animal, etc.) on a map, interact with objects and reach the final destination [7]. The currently most popular approach is to represent the single operations, that the virtual character can perform, in the form of Lego-like blocks that are assembled together to create a sequence (program) which solves the given task. A learner uses these blocks with intuitive drag-and-drop interactions in the programming environment and the blocks are shaped to indicate which blocks may be put together, in order to maintain the code syntactically correct [2, 7].

Block-based programming environments (BBPE) have gained on popularity among educators and learners through various applications [1], e.g., Scratch, Code.org, Blockly, Karel-3D. However, conventional way of interaction with virtual objects on computer screen with mouse or touch limits the learner from getting immersed into the learning environment. With current growth in availability of devices for augmented or virtual reality, we see possibilities for making block-based programming more immersive than before.

In this paper, we present our first results with Cubely, a block-based programming environment in virtual reality, where blocks are represented by cubes. The learner manipulates these cubes and assembles them together to solve programming exercises similar to those found in the Code.org e-learning platform. For the first evaluations, we templated the Cubely programming environment to a theme of the popular Minecraft[1] video game to further engage the learner. However, instead of assembling the source code on the computer screen with conventional Scratch blocks and seeing results in virtual environment, like in Microsoft Code Builder for Minecraft Education Edition[2], the program is assembled in the same environment where the code is executed and the learner does not have to switch between real and virtual reality. Other advantages of Cubely against existing BBPEs are:

- Learner can interact with cubes using the controllers in both hands simultaneously.
- The program can be placed anywhere in the motion space.
- Learner is located in virtual reality environment where he or she is not distracted by other surroundings.

2 RELATED WORK

Most of the existing approaches focus on easing the transition from blocks to text-based programming. Kölling et al. [5] have

VRST'2017, El Gothenburg, Sweden
2017. 123-4567-24-567/08/06…$15.00
DOI: 10.475/123_4

[1] https://minecraft.net/
[2] https://education.minecraft.net/trainings/code-builder-for-minecraft-education-edition/

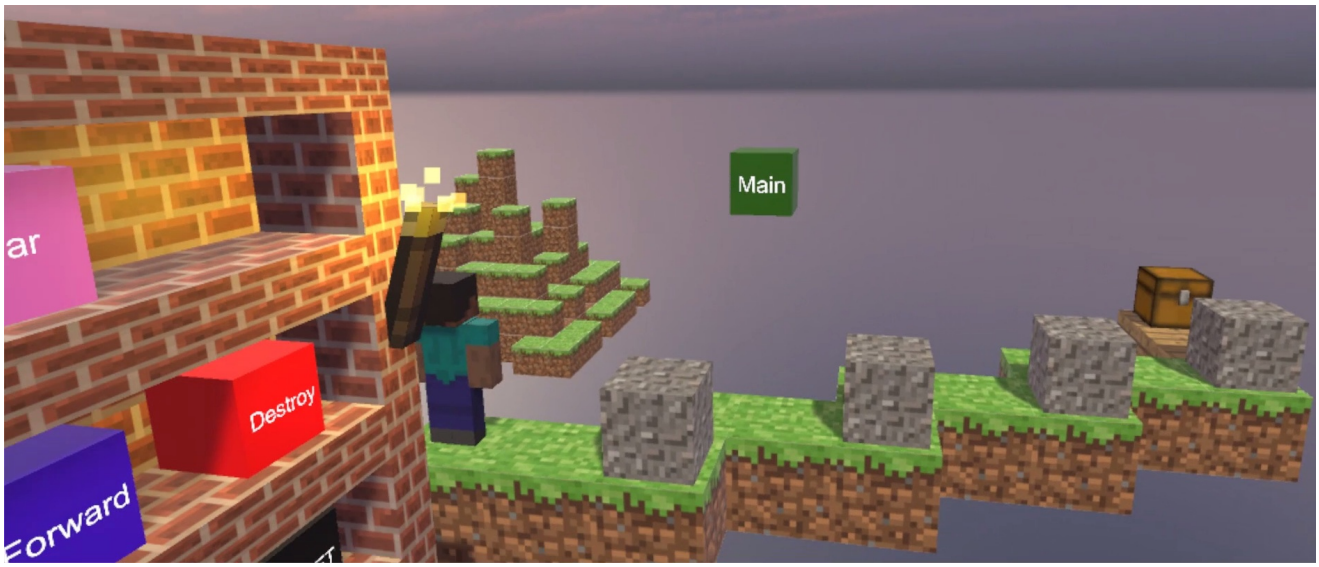

Figure 1: State of the world consists of (1) current state of the problem, (2) current program, and (3) player's motion space.

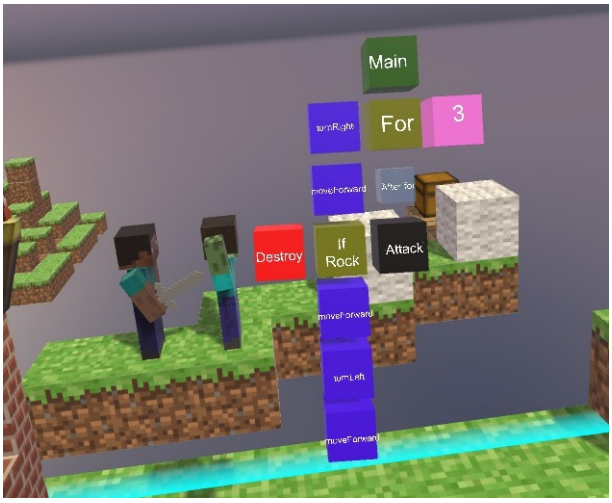

Figure 2: Program that uses a loop and a conditional to solve a challenge is being executed.

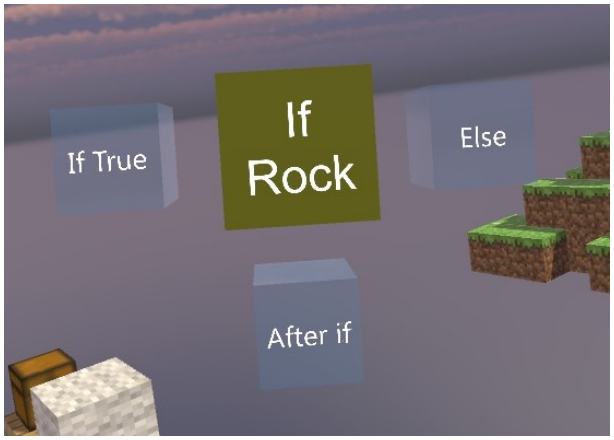

Figure 3: Display of placeholders of the *If Rock* cube.

proposed new frame-based editing mode that combines features of block-based and text-based programming. Frames are very similar to blocks, but crucially, keyboard entry is supported. Depending on the position and state of the cursor, keyboard input could be interpreted as commands (corresponding frames are inserted) or as traditional string input. Moreover, frames could be selected and manipulated just as text (cut/copied/pasted).

Bau [2] has developed Droplet, a new block-based editor used in Bau's Pencil Code [3] environment and App Lab at Code.org [4]. Droplet is a programming editor that allows dual-mode editing in blocks and text for any text program. This provides many of the advantages of blocks (e.g. less typing, assistance with syntax, visualization of the program) while allowing students to use text language runtimes and participate in text language communities.

Although these approaches provide more efficient way of interaction with block-based representation of source code, they ignore possible affordances of emerging virtual and augmented reality technologies such as immersion.

To immerse a learner into programming with blocks, Melcer [6] proposes tangible programming approach, where the blocks are real cubes that can be assembled together and the resulting algorithm is evaluated and displayed using the augmented reality. This way the learner changes behavior of a virtual character by interacting with real objects, what seems to be more natural for children than mouse device and a computer screen. However, learner may easily get distracted from the surroundings and the number of blocks from each type is limited.

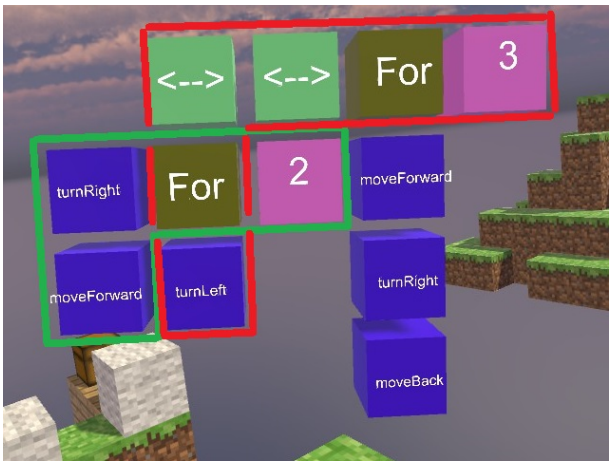

Figure 4: Nested loops with auxiliary *Link* cubes that enable the cubes in the nested loop to be laid out correctly. Red and green lines for clarity, they are not part of the VR environment.

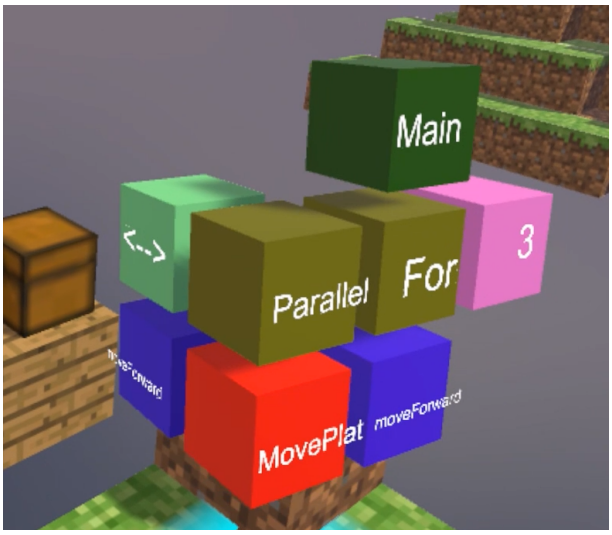

Figure 5: Example of a parallel branch.

3 CUBELY

To make block-based programming more immersive than before, we present Cubely[3], which sets the BBPE into sandbox world in virtual reality and represents the program blocks by cubes.

Figure 1 shows an initial state of the game world which consists of three main components: (1) current state of the problem, (2) current program, and (3) player's motion space. The problem is represented in a three-dimensional sandbox world based on the Minecraft video game and is viewed from the third person's perspective. The objective is to move the player's character from the initial position to the treasure chest and overcoming challenges along the way by the means of executing suitable program instructions. Challenges include white cubes that may contain either a rock or a zombie. The program used to operate the player's character is assembled within the same virtual reality environment using cubes that represent program instructions. The motion space to which the learner (player) is confined during the game contains prototypes of cubes that can be infinitely times used in assembling a program.

Programs are cube structures manipulated using the hand VR controllers. They can be placed anywhere within the VR environment and can be freely moved and rotated. The program's cube structure may consist of multiple 2D layers, each layer corresponds to a single thread of execution. Within a single layer, instructions that execute in sequence are laid out vertically below each other while compound instructions such as loops, conditionals that require a separate body (loop) or multiple bodies (true and false branches of conditionals) lay out the instructions in body horizontally to left or right side.

Figure 2 shows a simple program during its execution. Control flow is indicated by coloring the executed cube with black color. Execution starts in the *Main* cube, control flow moves to the cube below. The *For* cube executes its body 3 times, the number of iterations is specified using the number cube to the right of the loop's base cube. The body of the loop is to the left of the loop's base cube. It executes the instructions in sequence from top to bottom, with the exception to the conditional's base cube that will first evaluate the base cube (*If rock*) and then execute one of its branches depending on the base cube's outcome. Consequently, the program will solve the puzzle (see Figure 1) regardless of the contents of the hidden (white) cubes in the environment.

Cubes can be anywhere in the VR environment, and they can be grouped together in fragments. All cubes in a single fragment are aligned to a grid on a 2D plane and are moved and rotated simultaneously. The grid is uniquely determined by the location and rotation of the fragment's root cube. The root cube may be the *Main* cube or the topmost cube in the fragment. Parallel execution can extend the grid to the third dimension (see Figure 5).

During program construction using the VR handheld controllers, the newly attached cubes are automatically snapped to the grid with the help of placeholder cubes, which are smaller cubes indicating a possible fixing location or slot e.g. next instruction in sequence, the number of iterations of a loop, condition's branches. Figure 3 demonstrates the various placeholders for the conditional *If Rock* cube. When a cube (or cubes when manipulating the whole fragment) is approaching the placeholder a red shadow cube appears indicating the location to which the cube will be fixed after release from the VR handheld controller.

Note, that using this design of cube structures that requires programmer to place the cubes near each other, we can easily get into a situation where cubes from various parts of the program visually collide. Figure 4 shows the auxiliary *Link* cubes that can be used to fix the cubes at greater distances away from each other.

The learner can execute the assembled program starting from the *Main* cube and reset the environment by manipulating auxiliary buttons: *Start* button (cube) to start the execution, and *Reset* button (cube) to reset effects of previous program execution. Without triggering the *Reset* button between program executions, the learner can in effect solve the programming challenge iteratively using

[3]YouTube link to the video demonstration omitted to keep the submission anonymized. Please, see the video file enclosed with the submission.

small partial programs without ever completing a single program that fully solves the initial case.

4 EVALUATION

Cubely has been implemented in Unity engine to ensure compatibility with various VR devices.

In evaluation we mainly examined: (1) usability of manipulating cubes to build programs effectively, and (2) students' perceptions of VR programming using cubes in comparison to the traditional 2D block programming.

We evaluated Cubely in two observational studies with a total of 19 participants with average age of 17 years ($s = 2.97$), of which 8 were female (secondary education) and 11 male (2 with primary education, 8 with secondary education, and 2 with higher education). In each of the studies, participants first solved a series of tasks on Code.org website that uses Blockly, and afterwards the participants engaged in playing 9 levels in Cubely using HTC Vive immersive room-scale technology running on a workstation equipped with GeForce GTX 1080 video card.

The participants played six levels on Code.org such that they correspond to the levels played in Cubely except the final level that used the *Parallel* cube. Levels 1 and 2 in Cubely served as a tutorial on moving and turning the player's character in the game world. Levels 3 and 4 required to use simple cycles, Level 5 contained a simple obstacle, Levels 6 and 7 contained fog (white cubes what may contain either a rock or a zombie), Level 8 was the most challenging level that required the use of all preceding cube types. Level 9 was a simple level to use the *Parallel* cube.

At the end of the session, participants filled out a questionnaire that consisted of 5 questions, of which Q1-Q4 were answered on scale ranging from 1 (very negative) to 5 (very positive):

- *Q1 – Would you appreciate Cubely in courses at your school?*
- *Q2 – How challenging was it for you to solve the tasks?*
- *Q3 – How good was the provided way of interaction?*
- *Q4 – What is your overall impression?*
- *Which application do you prefer?*

Table 1 summarizes the responses (mean and standard deviation). Participants would appreciate Cubely more than Code.org for learning programming at their school. Solving tasks appeared to be less challenging for them in Code.org environment. In addition to this questionnaire, they praised ability to interact with cubes with both hands simultaneously. Study participants had not got prior experience with interaction in the VR environment and using the HTC Vive controllers. This slightly increased the time needed to learn the interaction with the environment and its objects. In the end, most of the study participants preferred Cubely over Code.org for the immersive experience, easier interaction with cubes with hands than blocks with mouse pointer, and not forgetting the setting in familiar Minecraft universe.

Table 1: Summary of questionnaire responses on scale ranging from 1 (very negative) to 5 (very positive).

	Q1	Q2	Q3	Q4	Prefer
Code.org	4.21 (s = 0.95)	2.11 (s = 1.02)	4.21 (s = 0.69)	4.16 (s = 0.67)	1
Cubely	4.74 (s = 0.71)	2.47 (s = 0.94)	4.53 (s = 0.68)	4.74 (s = 0.55)	18

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed a 3D approach to block-based programming that allows learners to create programs in virtual reality environment by assembling provided cubes (blocks) into structures. These programs control objects present in virtual scene and the learner's objective is to create a sequence which solves the given task. To effectively utilize the third dimension, learners can stack cubes along it in order to define parallel executions.

We evaluated our approach in two observational studies with a total of 19 students. They found it easy to work with the new VR interface and preferred it over the traditional 2D block-based programming.

In future work, we would like to design unique shape for each type of control flow primitives and add option to create, use and control data flow in assembled program. We plan to extend our evaluation within introductory programming courses during regular classes, and also to evaluate Cubely in our laboratory equipped with VR eye tracking and EEG technology to further analyze learner's behavior.

REFERENCES

[1] Efthimia Aivaloglou and Felienne Hermans. 2016. How Kids Code and How We Know: An Exploratory Study on the Scratch Repository. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, New York, NY, USA, 53–61. DOI: http://dx.doi.org/10.1145/2960310.2960325

[2] David Bau. 2015. Droplet, a Blocks-based Editor for Text Code. *J. Comput. Sci. Coll.* 30, 6 (June 2015), 138–144.

[3] David Bau, D. Anthony Bau, Mathew Dawson, and C. Sydney Pickens. 2015. Pencil Code: Block Code for a Text World. In *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*. ACM, New York, NY, USA, 445–448.

[4] D. A. Bau. 2015. Integrating Droplet into Applab – Improving the usability of a blocks-based text editor. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*. IEEE, 55–57.

[5] Michael Kölling, Neil C. C. Brown, and Amjad Altadmri. 2015. Frame-Based Editing: Easing the Transition from Blocks to Text-Based Programming. In *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE '15)*. ACM, New York, NY, USA, 29–38.

[6] Edward Melcer. 2017. Moving to Learn: Exploring the Impact of Physical Embodiment in Educational Programming Games. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*. ACM, New York, NY, USA, 301–306.

[7] David Weintrop and Uri Wilensky. 2015. To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. In *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*. ACM, New York, NY, USA, 199–208.