EPFL

CHILI
computer-human interaction
in learning and instruction

LEARN
Center for Learning Sciences

Create a block-based programming game in virtual reality

Author:
Nicolas Vial

Supervisor(s):
Richard Lee Davis

Professor:
Pierre Dillenbourg

January 5, 2023

Contents

1 Introduction

As the demand for computer programming skills grows, so too does the need for effective and engaging methods of teaching and learning programming concepts. One approach that has gained popularity in recent years is block-based programming, which allows even those with no prior coding experience to easily create and execute programs. With the advent of virtual reality and its growing use in education, there is a new opportunity to explore the potential of block-based programming in this immersive and interactive environment. In this study, we have created a block-based programming game in virtual reality using the Unity platform and Oculus Quest 2 headset. Through a series of user testing sessions, we iterated on the design of the game and collected data on how virtual reality affects the learning of fundamental programming concepts.

The purpose of this study is to compare the pros and cons of using virtual reality versus traditional 2D block-based programming applications. We aim to gain a deeper understanding of the potential of virtual reality as a teaching and learning tool in the field of computer science education.

The structure of this report is as follows: In the literature review, we will explore existing research on block programming in education using or not virtual reality. Next, we will describe the materials and methods used in our study, including the system design and user testing process. In the results section, we will present the data collected from the user testing sessions and discuss our findings. Finally, we will conclude with a summary of the key takeaways from our research and suggest directions for future work in this area.

2 Literature Review

2.1 Text-based and block-based programming languages for beginners

Text-based programming languages, also known as "source code" languages, are the traditional method for writing programs. These languages use plain text to represent instructions and commands, and are written in a specific syntax that the computer can understand and execute. Examples of text-based programming languages include C, C++, Java and Python. While text-based languages are powerful and versatile, they can be difficult for beginners to learn and use, as the syntax can be complex and it can be challenging for beginners to understand how to write programs using these languages.

In recent years, there has been a trend towards using block-based programming languages to teach programming to beginners. These languages use visual blocks of code to represent instructions and commands, which can be dragged and dropped into a workspace to create a program. Block-based programming languages are often easier to understand and use than text-based languages, making them well-suited for beginners. Examples of block-based programming languages include Scratch, Blockly and Code.org, which are widely used in educational settings and have been shown to be effective in helping beginners learn how to write programs[1] .

Overall, both text-based and block-based programming languages have their own unique features and benefits. Text-based languages are generally more powerful and versatile, but can be more difficult for beginners to learn and use. Block-based languages, on the other hand, are easier to understand and use, but may not have the same level of power and flexibility as text-based languages.

2.2 Programming in virtual reality

In the domain of VR, block-based programming can be particularly useful due to the immersive and interactive nature of VR technology. This can make it easier for beginners to learn programming, as they may be able to "walk through" their code and see how it works in a virtual environment, rather than just reading through lines of text on a screen.

There have been a number of efforts to explore block-based programming in VR, including Hack.VR[2], VR-OCKS[3], and Cubely[4], among others. These studies have shown promising results in the use of VR for block-based programming, but the literature in this area is still relatively limited.

To date, much of the research in this area has remained very close to existing 2D applications, exploring classical top-down and all-at-once approaches. However, there is an opportunity to explore new and innovative types of block-based programming that take advantage of the unique affordances of VR technology, such as embedded representations that leverage the unique possibilities of VR.

Overall, the use of block-based programming paradigms in VR offers the potential for an intuitive and interactive programming environment, and offers opportunities to explore new and innovative types of block-based programming. Further research is needed to fully understand and realize the potential of this approach.

3 Material and Methods

As explained before, the goal of the project was to develop a VR game using the Unity platform that would help beginners learn the basics of programming through a block-based game. The game, called "CodeBlocks Dungeon" takes place in a dungeon where the player must help a little robot called Bobi navigate through the levels and find his way out.

The player does this by giving instructions to Bobi using blocks of code. In order to be able to test and analyze a maximum of different parameters to capture the best assets brought by VR as well as to obtain the best parameters for an optimal use and learning by beginners in programming, we have developed 3 different game modes explained later. We finally concluded this project with a series of user tests in order to first perfect the game and then to analyze and draw conclusions on the best conditions of use for a beginner to learn in the best conditions.

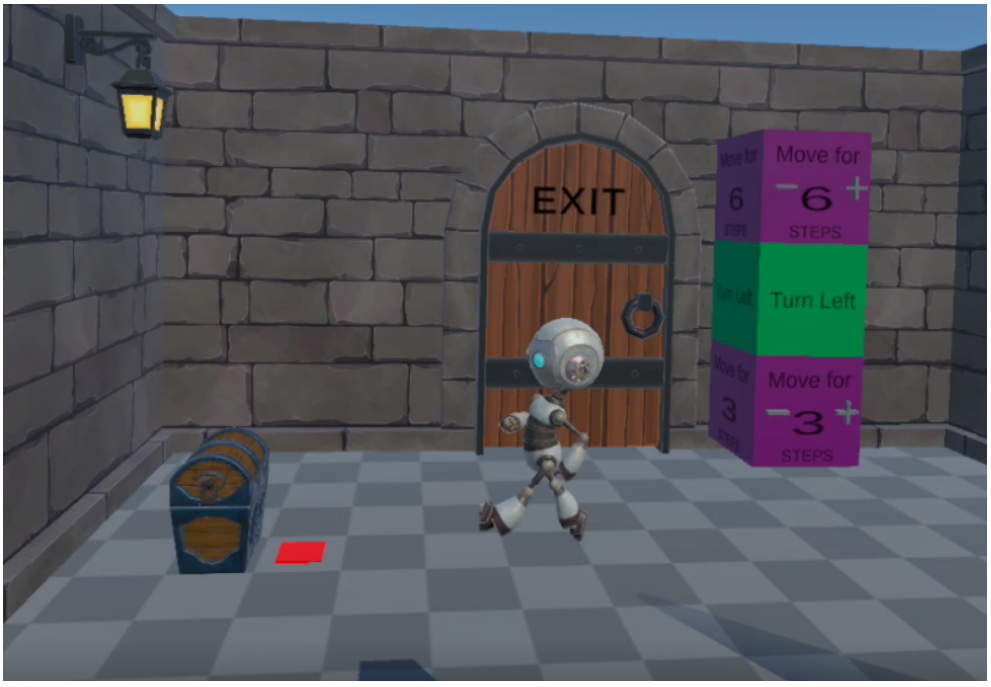

Figure 1: Image of the game where we can see the robot Bobi moving thanks to the code blocks placed by the player.

3.1 System Design

3.1.1 Interpreter

In designing the block-based programming language for our VR game, we considered a number of different approaches. Typically, the process of creating a programming language involves creating a lexer to identify the individual tokens in the code, a parser to produce an abstract syntax tree (AST) representing the structure of the code, and an interpreter to execute the code represented by the AST.

However, given the nature of our block-based language and the capabilities of the Unity platform, we decided to take a different approach. Rather than parsing and interpreting text-based code, we designed our interpreter to directly interpret physical blocks in the game.

Figure 2: Image of the game where all the blocks below the "Start block" in green are interpreted and executed in-order to make Bobi move first from 6 step, then to turn left and finally to move for 3 steps.

When a block is activated (e.g. by pressing a "start" button or by being the next instruction), the interpreter simply identifies the type of block (e.g. movement, operation, etc.) and executes the appropriate actions in the game. We then adapted this general interpreter system to the different game modes we developed.

3.1.2 Language designs

In building the different game modes for our block-based programming language, we considered a number of different parameters. The first parameter was the sense of execution, which refers to the order in which the code is executed. We included both top-down and bottom-up options, with top-down execution starting at the top of the code and working its way down block by block, and bottom-up execution starting at the bottom of the code and working its way up block by block.

Figure 3: Image of the game in the third game-mode where all the blocks are executed in the bottom-up order. The "Move for 6 steps" is thus executed first.

The second parameter was the abstraction type, which refers to how the code is integrated into the game environment. We included both embedded and non-embedded options, the embedded option requiring the player to place the code directly on top of Bobi or on his path, and the non-embedded option allowing the player to place the code anywhere in the game environment but removing any interaction with Bobi.

Figure 4: Image of the game in the second game-mode where all the blocks are executed in the top-down order. The abstraction type is embedded since the blocks are placed above Bobi and not below a "Start block" as in the non-embedded abstraction.

Based on these two parameters, we also included a third parameter: the construction method, which refers to the way in which the code is built. We included both all-at-once

and split options, with the all-at-once method requiring the player to build the entire code in a single chunk of blocks, and the split method (available only for the embedded abstraction) allowing the player to split the code into multiple blocks and place them on Bobi's path.

Figure 5: Image of the game in the second game mode where the blocks have been split. The first block is executed from the beginning and the two blocks on Bobi's path are executed once Bobi passes underneath and has finished all his previous moves.

Using these parameters, we developed three game modes for the game:

1. The first game mode is a top-down, non-embedded option with an all-in-one construction method. This game mode is designed to be similar to traditional block-based programming languages such as Scratch but implemented in 3D.

2. The second game mode is a top-down, embedded option with a split construction method. This game mode is designed to be more "embedded" into the game and allow players to get a more immersive and active experience. By splitting the code into multiple blocks and placing them in Bobi's path, players can get a better sense of how their code is executed and the effect it has on Bobi's movements.

3. The third game mode is the same as the second game mode, but with a bottom-up execution order.

Overall, these three game modes allowed us to explore a range of different options for teaching block-based programming in VR, and provided players with the flexibility to choose the approach that best suited their learning style and preferences.

3.1.3 Level and game design

In virtual reality, level and game design are crucial components of the overall player experience. A well-designed game and level can help to create a sense of immersion and engagement, while a poorly designed game and level can break that immersion and lead to frustration or disinterest. Therefore, it is important to carefully consider the design of these elements in order to create a successful VR game.

In the block-based programming game "CodeBlocks Dungeon," the game is set in a dungeon where a small robot named Bobi has become lost. The player's goal is to help Bobi escape from the dungeon by guiding him through a series of levels, each of which is a separate room with a chest that Bobi must reach. To guide Bobi through the levels, the player must use blocks of code representing different actions, such as movement commands (e.g. move forward, turn left/right) and operation commands (e.g. repeat). The player must build the code by arranging the blocks in the desired order and then activating it to execute the instructions.

Figure 6: Image of the game during the third level. We can see all the available blocks being executed in order to solve the level. The "repeat until chest" block repeats the inside instructions indefinitely until Bobi reaches the chest.

In designing the levels and game-play for "CodeBlocks Dungeon," we made a number of specific choices to enhance the player's immersion and enjoyment. For example, we

designed Bobi to be a cute and likable character in order to help the player become emotionally invested in his journey and more immersed in the game. We have also included a variety of challenges and obstacles such as limiting the number of usable blocks and increasing difficulty as the levels progress to keep the game-play interesting and engaging.

Overall, the level and game design of "CodeBlocks Dungeon" is designed to be immersive and enjoyable for players of all skill levels, while also providing a challenging and rewarding learning experience. By carefully considering the design of these elements, we aimed to create a successful and engaging VR game that will keep players coming back for more.

3.2 User Testing Process

Once the first prototype of the game was completed, user testing was conducted to assess the usability and effectiveness of the game. A total of nine people participated in the testing, with the game being updated and iterated based on feedback five times. The testing helped to improve the overall design and user-friendliness of the game, and provided valuable insights into the preferences of players. During the user testing process, we asked participants to complete a series of tasks (See complete interview protocol in Appendix 6.1). These tasks included finishing the tutorial and the first level in the first game mode, finishing the tutorial and the second level in the second game mode, and finishing the tutorial in the third game mode. We also asked participants to choose their preferred game mode to complete level 3.

Once the tasks were completed, we asked participants a series of questions to gather data on their programming experience and familiarity with VR. We also asked participants to share their preferred game mode and their reasons for this preference, which helped us to understand whether players preferred top-down or bottom-up execution, all-at-once or split code, and embedded or non-embedded representations. In addition to these questions we asked a number of more oriented questions in order to gather feedback on the game-play, level design, and overall player experience, and to identify any issues or areas for improvement. Based on the feedback we received from the user testing, we made a number of updates and improvements to the game, including adjusting the difficulty of the levels, improving the controls, adding new features and content and solving bugs that break the game and the immersion.

The user testing process was a crucial aspect of the development of the block-based programming game. It not only enabled us to gather valuable feedback and make necessary improvements to the game, but also provided us with a range of results that allowed us to draw conclusions about the most effective ways to teach programming concepts using a block-oriented programming game in VR. These findings are discussed in more detail in the following chapter. Overall, the user testing process played a vital role in creating a more enjoyable and effective game, and in understanding the benefits of using VR for learning programming through a block-based game.

4 Results

The goal of our project was to explore a new way for learners to engage in computer programming, using a block-based programming game in VR. We had several hypotheses about the potential benefits and challenges of using VR for this purpose, and in the following results section we will discuss the evidence we found from user testing that supported or did not support these hypotheses. Additionally, we will report on any interesting findings that emerged from the user testing process that we did not expect.

The results section is divided into two parts. The first part discusses the specific learning and improvements we made to the block-based programming game based on user testing, and the second part presents more general findings about programming in VR based on the user testing data.

4.1 Iterations on the application based on user testing

During the user testing process, we made several iterations and improvements to the block-based programming game based on the feedback and observations we gathered from participants. The first batch of user testing was primarily focused on fixing breaking bugs in the game, which made it difficult to gather insights on other aspects of the game. However, we were able to identify and fix the main bugs, enabling us to conduct more thorough testing in subsequent batches.

One of the key learnings from the user testing was the importance of offering a range of options for movement in VR. The immersive nature of VR technology can make it more enjoyable for players, but it can also increase the risk of cybersickness. To address this, we initially implemented teleportation as a fast and smooth way to move around in the game. However, some players preferred more realistic smooth motion using a joystick. Similarly, some players found jerky rotation to be intuitive, while others found it confusing. To address these preferences, we added multiple options for movement, allowing players to choose their preferred method.

Another important finding was the need to clearly explain the controls and game-play to players. In the first iteration, we relied on text explanations, but many players found it inconvenient to read while standing in VR and there were a number of misunderstandings as a result. We therefore revised the explanations to include more images and drawings, which were more easily understood by participants.

The game design was also a key factor in the enjoyment and success of the game. Participants consistently commented on the music and the cute design of the character, Bobi, and stated that these elements contributed significantly to the overall enjoyment of the game. In particular, the immersive nature of VR technology allowed us to create a more engaging and interactive experience for players.

Finally, we found that the level of difficulty was a major factor in the success of the game. While some players found the final level to be too difficult, others found it to be relatively easy. This difference appeared to be related to the players' familiarity with VR and the game-play mechanics. Unfortunately, we did not have time to implement a wider range of difficulty levels, but this would be a valuable improvement to consider in future

(a) textual explanations

(b) explanations via drawings

Figure 7: The first image shows the explanations during the first test iterations which were then modified into drawings to make them faster and easier to understand.

iterations of the game. By offering a range of difficulty levels, we could cater to the needs of different types of learners and ensure that the game is accessible and enjoyable for all players.

4.2 More general findings based on user testing

In this section, we present the more general findings based on user testing of our VR programming game. We address five important research questions and compare our hypotheses to the data obtained during the tests.

Tester Number	Has done programming before	Has done VR before	Preferred sense of execution	Preferred construction method	Preferred abstraction
1	Yes	Yes	Top Down	All-at-once	Nonintegrated
2	Yes	Yes	Top Down	All-at-once	Nonintegrated
3	No	No	Top Down	All-at-once	Nonintegrated
4	No	No	Top Down	All-at-once	Embedded
5	No	Yes	Bottom up	All-at-once	Embedded
6	Yes	No	Top Down	All-at-once	Embedded
7	No	No	Bottom up	All-at-once	Embedded
8	Yes	No	Bottom up	All-at-once	Embedded
9	No	Yes	Top Down	All-at-once	Embedded

Figure 8: Results of the research questions asked to the participants during the user testing.

The first research question concerned the sense of execution of the code between top-down and bottom-up. Out of the nine participants, 6 chose the top-down sense of execution, citing its intuitive nature as it looks like a list and reads in the same direction as

the reading direction. Two of these participants also referenced the programming sense of execution. Meanwhile, the three participants who chose the bottom-up sense included two who had never done programming before. These two used the building order in construction games such as LEGO, which goes from bottom-up, as their argument. The third participant, who had prior programming experience, argued that the bottom-up sense was more intuitive in the embedded game mode because they placed the blocks above Bobi and the first block to be executed had to be the one closest to him. This data suggests that there may not be an obvious choice for the sense of execution for those new to programming, but since the sense of execution of typical code is top-down, it may be more sensible to select this sense of execution. It would be valuable to find a way to make the reading direction of the blocks more obvious and intuitive by using visual tools such as visible numbering of the blocks in the execution order.

The second research question focused on the possibility of splitting the code to take advantage of the strengths of VR. The results were in total contradiction with our hypothesis, as all participants were unanimous in their preference for the "all-in-one" construction method, i.e. despite the fact that users had the possibility to integrate blocks of code into the scene, they simply chose to build a single stack and place it above Bobi's head. They argued that it was much easier to simply build a stack of blocks rather than splitting them and needing to move them to the correct location. However, some participants found the idea of splitting the blocks to be interesting and thought it could be useful in more challenging levels where it becomes difficult to make the entire path in one stack of blocks. From these comments, we believe that the idea of splitting should not be discarded but rather further explored and tested to determine its utility and relevance in VR.

The third research question concerned the abstraction between embedded and not embedded modes. The first three testers chose the non-embedded mode, while the last six testers chose the embedded mode. This difference is likely due to changes made based on feedback from the first two batches of tests. In the initial tests, players had to place the blocks inside Bobi rather than above him, which was cumbersome and fatiguing. Once this issue was addressed, players preferred the embedded game mode because it improved immersion with Bobi in the dungeon rather than being external as in the non-embedded mode. This result met our expectations and demonstrates the importance of addressing user feedback to improve the VR experience.

For the fourth research question, we asked participants if they had prior experience with programming. If they did not have any experience, we asked them if they learned anything about programming from playing the game. The most common responses were related to the idea of defining and then executing code in order to accomplish a specific task. For those who did have prior programming experience, we asked them in what ways they thought the game could be helpful for beginners learning programming and in what ways they thought it might not be as helpful. The majority of respondents believed that the game could help beginners understand the concept of stacking and executing instructions, as well as introduce fundamental programming concepts such as for loops. However, some users pointed out that elements such as the 3D interface, gamification, and the split code feature might not be as closely aligned with traditional programming

practices. the answers for this question were not clear and identical for all users, preventing us from drawing interesting conclusions.

The final research question focused on comparing the VR game to a 2D version of block-based programming. Before asking this question, we showed users a 2D version to ensure they were familiar with it. The responses to this question were consistent across all participants, with all of them stating that it would be easier and faster to play in 2D, but that the VR version was more enjoyable, immersive, and engaging as we expected.

5 Conclusion

In conclusion, our user testing showed that there are mixed opinions on the sense of execution for block-based programming, with some participants finding top-down more intuitive and others finding bottom-up more intuitive. However, the majority of participants preferred the top-down sense of execution, which aligns with the traditional sense of execution in programming.

The possibility to split code in order to use the strengths of VR was not as successful as expected, with participants uniformly preferring the all-at-once construction method. However, some participants did express interest in the idea of splitting blocks for harder levels, suggesting that this feature may be worth further exploration and development.

The abstraction between embedded and non-embedded modes was well received, with the majority of participants preferring the embedded mode due to the improved immersion it provided.

Finally, the participants found a real interest in this VR version of the block programming game compared to the 2D version, finding it more enjoyable, immersive and stimulating, largely due to the different aspects of game design that were much less present and enjoyable in 2D.

Overall, the system met many of our expectations but there are still areas for improvement, such as finding a way to make the sense of execution more intuitive for those new to programming and considering the utility of the split code feature. As this game is only a first version, it would be interesting to improve it by adding more blocks related to programming such as if/else, while or for loops and to conduct user tests to analyze the evolution of the results. Another interesting idea would be to use the possibility of playing with two people at the same time in VR and thus have levels played by two people or simply to allow teachers to help students more easily. Further research and testing could also be conducted to explore the benefits and drawbacks of the VR format for block-based programming and to address any remaining questions or concerns.

References

[1] Yoshiki Tanaka Yoshiaki Matsuzawa and Sanshiro Sakai. "Measuring an impact of block-based language in introductory programming". In: (Jul 2016). DOI: 10.1007/978-3-319-54687-2_2.

[2] Dominic Kao et al. "Hack.VR: A Programming Game in Virtual Reality". In: (2020). DOI: https://doi.org/10.48550/arXiv.2007.04495.

[3] Rafael J. Segura et al. "VR-OCKS: A virtual reality game for learning the basic concepts of programming". In: (2019). DOI: https://doi.org/10.1002/cae.22172.

[4] Juraj Vincur et al. "Cubely: Virtual Reality Block-Based Programming Environment". In: (2017). DOI: https://doi.org/10.1145/3139131.3141785.

6 Appendix

6.1 Interview protocol

6.1.1 Protocol

1. Explain the application to the participant
   (a) Oral explanations of the general principles of the game
2. Start with the non-embedded, top-down environment (first game-mode)
   (a) Complete the tutorial
   (b) Complete Level 1
3. Switch to the embedded, top-down environment (second game-mode)
   (a) Complete the tutorial
   (b) Complete Level 2
4. Switch to the embedded, bottom-up environment (third game-mode)
   (a) Complete the tutorial
5. Play Level 3 in the environment of their choosing
6. Short interview

6.1.2 Interview questions

1. Have you ever programmed before?
2. Have you ever used VR before?
3. Usability issues (difficulties with controllers, difficulties placing and moving blocks, etc.)
   (a) I saw that you were having some difficulties with ____, can you explain a bit what was happening there?
4. Conceptual issues (difficulties writing a working program, confusion about direction of code flow, etc.)
   (a) I noticed that you were having trouble with ____, can you explain a bit what was happening there?
5. Level design / Game design
   (a) Did you feel like the puzzles were at the right level of difficulty?

   (b) Did you enjoy working on the puzzles or was there something about them that was not so fun?

6. Comparing the two programming styles
   (a) Which programming style did you prefer (linear or embedded)?
      i. Can you explain?
7. Learning
   (a) If you haven't programmed before...
      i. Do you feel like you learned something about programming? If so, what did you learn?
   (b) If you have programmed before...
      i. What aspects of introductory programming do you feel this could help people learn?
      ii. What aspects of introductory programming do you feel this is not helping people learn?
8. VR specific
   (a) Can you imagine using a 2D version of this application? (Show an image of a Blockly maze)
   (b) What do you think doing this in VR adds to the learning experience?
9. Improvements
   (a) What would you change about this application?

6.2 Source code

The code of this project can be found on the following repo: https://github.com/NicolasVial/Block-based-programming-game-in-VR-CHILI-LAB

More precisely, the scripts of the different game modes can be found in the "BlockBasedLanguageProject" folder (in VR/Assets/).