2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW) | 978-1-6654-4057-8/20/$31.00 ©2021 IEEE | DOI: 10.1109/VRW52623.2021.00192

Programmable Virtual Reality Environments

Nanlin Sun* Virginia Tech, USA
Annette Feng† Virginia Tech, USA
Ryan Patton‡ Virginia Commonwealth University, USA
Yotam Gingold§ George Mason University, USA
Wallace Lages¶ Virginia Tech, USA

ABSTRACT

We present a programmable virtual environment that allows users to create and manipulate 3D objects via code while inside virtual reality. Our environment supports the control of 3D transforms, physical, and visual properties. Programming is done by means of a custom visual block-language that is translated into Lua language scripts. We believed that the direction of this project will benefit computer science education in helping students to learn programming and spatial thinking more efficiently.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality; Human-centered computing—Human computer interaction (HCI)—Interactive systems and tools—User interface programming;

1 INTRODUCTION

Programming and computational thinking skills are essential for success in many disciplines, including science and the arts. However, programming can be intimidating to many students. Learning to program involves not only an understanding of the language syntax and idiom but also how to translate problems and their solutions in a way that can actually be solved by a computer. Block-based languages (BBL) such as Scratch [4] became part of the computer science education landscape since they allow users to program by manipulating visual elements instead of text.

On the other hand, psychological aspects such as motivation, and self-efficacy are also important for student success. Virtual environments (VE) provide an opportunity for creating rich and immersive experiences that are more engaging than the ones from traditional learning environments. For this reason, virtual environments have been explored as a way to support learning in many different domains.

My Reality (MYR) is an application that combines programming with a virtual reality visualization [1]. VR was found beneficial for presenting abstract computer science concepts in a more tangible way for novice programmers. However, MYR requires users to use the code editor on a Web-Based platform outside VR to write the code. We believe that an all-in-one experience in VR would smooth the workflow (users don't have to switch back and forth between two platforms when making modifications to their current work).

Another VR coding environment is the 3D Virtual Programming Language (3D-VPL). Instead of using a 2D canvas, 3D-VPL uses tridimensional blocks to represent classes and objects. Users can move around and add or remove objects allowing them to visualize programming concepts [3]. Unlike 3D-VPL, our environment uses a true block language, and requires less user interaction on the manual code organization as the code lines grow large; intuitive user input and interaction in VR is also of great importance for coding efficiency and user experience (UE).

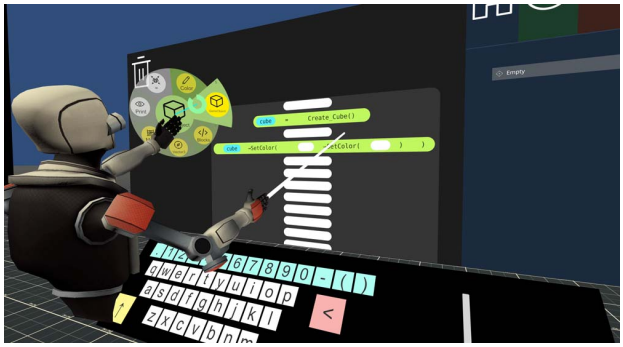

Figure 1: User avatar inside the programmable virtual environment

In our work, we built an all-in-one programmable VR environment with many features including the ability to write, execute, and pause, save and load code. To evaluate our application we developed an introductory curriculum that teaches users how to use the programmable VR environment and gain a general knowledge of programming. In this paper, we describe the technical and design elements as well as our initial impressions of the system.

2 VIRTUAL REALITY PROGRAMMING

Our VR programming environment consists of an integrated coding panel and menus for accessing and managing coding elements during programming. The two-dimensional code editor uses a visual block language inspired in Scratch to allow users to create blocks minimum user input (Fig. 1). Using code, the user can instantiate and manipulate objects in the same surrounding space. The user could run and stop the execution of the code using dedicated play and stop buttons in the 3D space. The environment was developed using Unity 3D [1] and the Lua programming language [2].

2.1 Architecture

Our project uses a three-tier architecture, where the implementation is separated into the presentation layer, the logic layer, and the data layer. The presentation layer manages the user input and the interaction techniques that allow the user to interact with the VR programming environment (code editor, pie menu, virtual keyboard, etc.).It takes direct signals from user interaction and instructs the logic layer to generate, compile, and execute the code. The logic layer manages all the code related operations including code generation, runtime code compilation, and execution. Execution happens by interpreting Lua code and calling the corresponding bindings in the Unity C# runtime. The data layer is responsible for managing both the memory instance of the code in the memory and serializes

*e-mail: nannie@vt.edu
†e-mail: afeng@vt.edu
‡e-mail: rpatton@vcu.edu
§e-mail: ygingold@gmu.edu
¶e-mail: wlages@vt.ed

[1] unity3d.com
[2] https://www.lua.org/

978-1-6654-4057-8/21/$31.00 ©2021 IEEE
DOI 10.1109/VRW52623.2021.00192

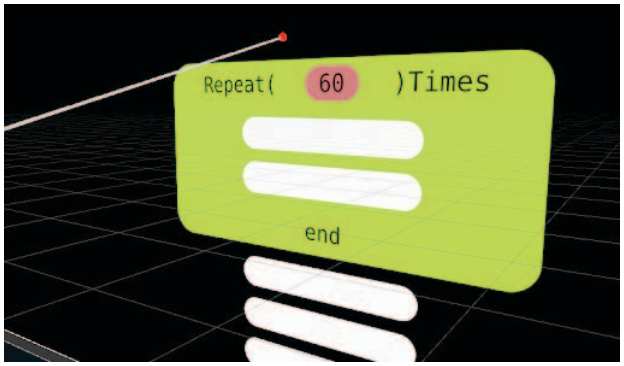

Figure 2: Loop block in the code editor. White lines indicate where new statements can be added.

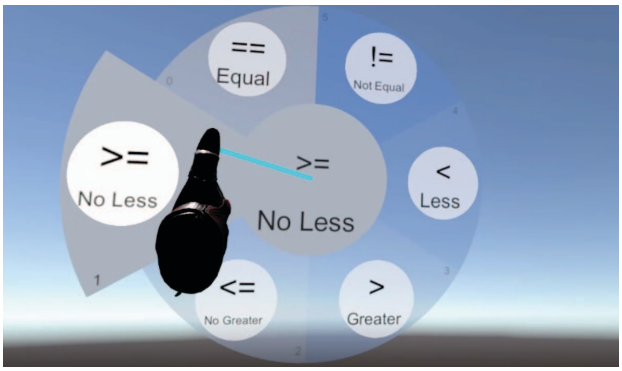

Figure 3: The User selects the “No Less” code block on the pie menu with the VR controller

the code to the local file system. The data layer converts the code to and from the serialized format.

2.2 Code Generation, compilation, and execution

The code generation is completed at runtime by the logic layer with the help of the VR code editor. The VR code editor allows users to write code by putting together code blocks. Each code block represents a single unit of the element in programming, including a single variable, a statement, and a code structure. The code editor generates the code by concatenating all the string values of the connected code blocks into a code script. The code script is then sent to the runtime compiler for compilation and execution.

Currently, our runtime compiler runs in MoonSharp, a Lua interpreter written entirely in C# for the .NET, Mono, and Unity platforms. It allows generated Lua Code to be executed on the fly. There are, however, noticeable differences between C# and Lua Code. For example, the lack of native support for coroutines and classes in Lua. Thus, in order to utilize most of the potentials and functionalities of C# at runtime, some of the wrapper functions and interfaces are set up manually in C# for the MoonSharp in advance.

2.3 Interaction design

In order to provide natural interaction in VR, all the interaction happens with six degrees of freedom controllers. Block selection was implemented with a hand oriented pie menu instead of the traditional palette used in Scratch (Fig. 3). Users activate the pie menu by holding the side button of the VR controller. The pie menu appears facing the users at the location of the VR controller when the side button was initially pressed. The user can then make the selection by hovering the VR controller over the pie menu option and then release the side button. A code block of the selection will be generated on the code editor and the user can use the controller pointer to relocate the code block to the desired position in the code.

To enter variable names and numeric values, the user can use a virtual keyboard below the code editor. Since controller pointing is still one of the best options for selection-based text entry in VR, the pointer is used to operate the keyboard, as well as for both block manipulation [2].

2.4 Curriculum

We built objectives in modules and curriculum for users to follow in order to teach them how to program in VR programming VE. We have modules that are focusing on the concept and usage of variables, conditional statements, loops, functions, algorithms, 3-D coordinate systems, vectors, programming environment block language, etc.

By going through each module, users will learn not only how to use the tools in our VE to code with proper input and interaction, but also different programming concept, syntax, and logic.

3 Discussion and Future Work

Our preliminary expert evaluation indicates that the virtual environment is easy to operate. Users are able to procedurally instantiate objects, move them in space, and change physical properties such as color and collision response parameters. Creative ideas, such as building infinite stairs made of small cubes and moving objects to simulate candy rainfall are also possible. However, large changes in the code are time-consuming, due to the need of dragging blocks with the pointer. In addition, even though the pie menu reduces the time necessary to create a new block, it is not as fast as the equivalent mouse and keyboard interface.

Due to the noticeable differences between C# and Lua Code, such as lack of support for coroutines and classes in Lua, some of the wrapper functions and interfaces had to be set up manually in C#. We plan to improve our runtime compiler by switching from MoonSharp Lua to Roslyn C# in order to utilize most of the potential of C#. We also plan to explore more options for block selection and manipulation, as well as to conduct a formal user study.

4 Conclusion

We have presented the design and implementation of a programmable virtual reality environment based on a block-based visual language. Our initial evaluation suggests that block languages are an intuitive way to do programming in virtual environments. We plan to continue to develop the interaction aspects to make large code changes easier and more efficient.

Acknowledgments

This research was funded in part by 4-VA, a collaborative partnership for advancing the Commonwealth of Virginia.

References

[1] C. Berns, G. Chin, J. Savitz, J. Kiesling, and F. Martin. Myr: A web-based platform for teaching coding using vr. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp. 77–83, 2019.

[2] P. Z. Marco Speicher, Anna Maria Feit and A. Krüger. Selection-based text entry in virtual reality.

[3] F. R. Ortega, S. Bolivar, J. Bernal, A. Galvan, K. Tarre, N. Rishe, and A. Barreto. Towards a 3d virtual programming language to increase the number of women in computer science education. In *2017 IEEE Virtual Reality Workshop on K-12 Embodied Learning through Virtual & Augmented Reality (KELVAR)*, pp. 1–6. IEEE, 2017.

[4] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. S. Silver, B. Silverman, et al. Scratch: Programming for all. *Commun. Acm*, 52(11):60–67, 2009.