DOI: https://doi.org/10.56198/ZH3JO8GAE

Work-in-Progress—iCoding: Immersive Coding in Unity

Manuela Chessa, Giorgio Delzanno, Davide Giovannetti, Giovanna Guerrini,
Filippo Manini, Davide Miggiano, Marianna Pizzo, and Eros Viola

DIBRIS, University of Genova, Italy
giorgio.delzanno@dibris.unige.it

Abstract. This work-in-progress paper presents a novel application of Virtual Reality (VR), via the Unity framework, in Computer Science Education to provide an immersive experience for Computational Thinking and Coding activities. First of all, we have created a 3D Unity game whose scenario consists of a game room with arcade custom cabinets. Players can move in the game room, select a cabinet by simply approaching it, and solve coding exercises implemented in a 2D arcade game projected on the cabinet's screen. The novelty of our 3D application is that each cabinet, besides providing a different arcade game, is equipped with a virtual block editor through which players can provide solutions to the proposed computational thinking and coding challenges. In a second step, we have ported the 3D Unity game to an immersive VR application via SteamVR, e.g., using tele-transport for player movement. Our framework transfers traditional coding activities (based on visual languages and arcade games) into an immersive experience in a VR scenario. This combination requires additional skills, such as rapidly adapting the passage from 3D to 2D scenarios during a game. In this paper, we describe the key feature of the application and the expected learning potential and outcomes.

Keywords: Computer Science Education, Computational Thinking, Immersive Environments.

1 Introduction

Virtual Reality (VR) technologies are becoming more and more accessible in terms of application development frameworks, usability, and visualization, e.g., head-mounted devices (HDM) costs. A wide range of frameworks exists for application development, from beginner level, such as CoSpaces to Unity [1] and Unreal. Usability has substantially improved thanks to a new generation of devices such as the Oculus Quest devices1. The cost of Oculus Quest 2 is almost comparable to those of off-the-shelf game consoles.
This scenario looks increasingly favorable to exploring applications of immersive Virtual Reality to design new forms of interactive learning activities. Learning through Virtual Reality may bring other benefits: it offers an infinite range of simulation scenarios, it allows designing new forms of hands-on activities, it enables the involvement of all five senses, and thus it increases learning compared to more passive methods [2].
A challenging research goal is to design new forms of immersive experiences in Computer Science Education and to enrich those approaches based on Computational Thinking and Coding [3]. Computational Thinking involves expressing problems and solutions in ways an executor could understand, e.g., see [3, 5]. Coding is the process of encoding solutions into the language of a computer. Coding languages are often simplified versions of real programming languages that provide visual block editors integrated with "WYSIWSG" animation GUI's.

Our project, born in the context of a Computer Science Education course for the Bachelor's degree in Computer Science at the University of Genoa, Italy, aims at transporting a series of coding activities, based on games, block languages, and reasoning, within an immersive environment. The intended outcome is to stimulate curiosity towards computational thinking and coding activities by increasing the sense of novelty and adventure related to

[1] https://www.meta.com/it/quest/products/quest-2/

the exploration of a virtual world and the level of attention and concentration when solving the proposed exercises, exploiting the sense of presence and engagement in VR.

To achieve this goal, we have developed an experimental immersive VR Unity application, whose scenario consists of a 3D game room with arcade custom cabinets. Players can move in the game room, select a cabinet (by simply approaching it), and then solve coding exercises implemented in a 2D arcade game projected on the cabinet's screen. Solving coding exercises makes the 2D arcade games work correctly.

The novelty of our 3D application is that each cabinet, besides providing a different arcade game, is equipped with a virtual block editor devised for this project (UEBlock), through which players can provide solutions to the proposed exercises. The 3D Unity game has been ported to VR via SteamVR (implementing player movement via teleportation and allowing interaction with the blocks through the HMD controllers).

The project is currently being tested in single-user mode. However, its extension to the multiplayer case would allow the creation of new and more engaging coding labs for groups and classes, that might be remotely connected.

In the paper, we present the main features of the proposed 3D/VR game and details related to its development.

2 The Virtual Arcade Game Room

The environment consists of a virtual arcade game room with four different cabinets each one presenting a different 2D arcade game, namely Albion, Legend of Carl, Dystopian Future, and Pathfinder. Each arcade game consists of a graphical game board and a set of coding problems to be solved via the visual programming language UEBlockly, specifically devised in Unity. Each 2D arcade game is projected on a cabinet's screen. Players interact with the cabinet via a virtual pad and a button, as shown in Fig. 1.

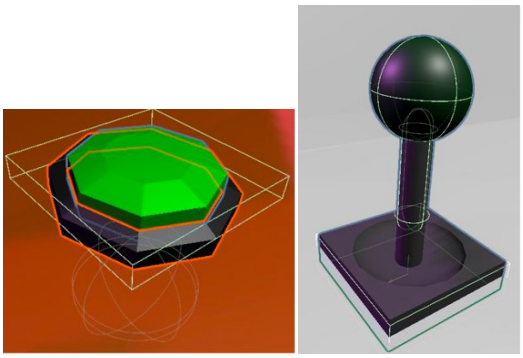

Fig. 1. Gamepad and button associated to each cabinet.

The four arcade games propose a series of coding problems (e.g., moving a sprite in a specified direction) that the player can solve by interacting with a virtual block editor for the UEBlockly language. In this way, the block editor can be used to create scripts to animate the arcade game sprites.

Players can also use a virtual keyboard to insert commands, create variables, and assign them names and values needed to complete the script (e.g., create a variable to set the speed of sprite movement, create a variable to enable a given behavior). In Fig. 2, we show one example of an arcade game created for the cabinet. The UEBlockly commands are designed to control the animation (e.g., direction and speed) of the sprite in the arcade game. This way, players can create solutions to the proposed exercises in the form of executable UEBlockly scripts.

Fig. 2. The Legend of Carl: an example of arcade game.

The realization of this application posed numerous technical and user experience challenges. Indeed, the project covers advanced aspects of the Unity framework related to the interaction between the user and virtual objects. During the virtual experience, the user must press buttons, grab, and move the virtual pad insert alphanumeric inputs (e.g., sprite attributes and values) via a virtual keyboard. In the current implementation, interaction is obtained through the HMD controllers, the selection is achieved through raycasting, and inputs are given by pressing the controllers' buttons.

The editor provides the following set of commands:

- The *CreateVar* block is used to create a new variable.
- *AssignByBlock* assigns a value to a variable.
- *GetVar* is a getter for a given variable name.
- *BoolOpBlock* defines a comparison between variables and values.
- *FloatOpBlock* defines expressions with arithmetic operators.
- The *If* and *While* blocks require a *BoolOpBlock* for the Boolean condition and implement alternative and iterations control flows.

In Fig. 3, we show a possible solution for the problem proposed in the Albion games: "Our character must jump to avoid falling into the hole. To make this possible, we need to change the speed value to 1 and, only then, enable the jump through the enableJump variable."

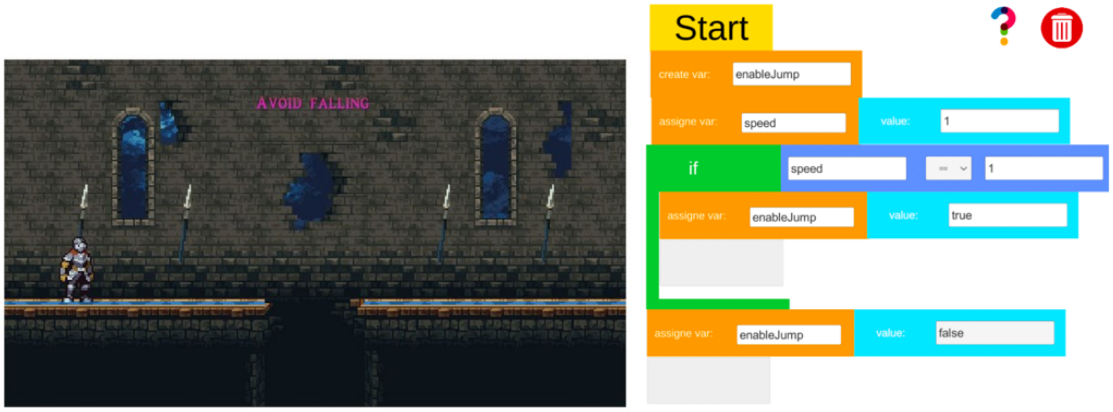

Fig. 3. UEBlockly script the user must write (right) to allow the character to jump and avoid falling into the hole (left).

When the user approaches one of the arcade games and starts playing with it, a message is shown explaining that a coding action is required to make the game work correctly. The UEBlockly editor is displayed on a canvas next to the cabinet, explaining the coding exercise to be solved. The exercise must be solved using the proper block, creating and assigning the correct variable values, and implementing the correct algorithm.

The design of the UEBlockly visual editor in Unity (a canvas with graphical widgets) required solving additional nontrivial issues related to the interaction between controllers and graphical widgets, e.g., moving a block from the block palette to the editor, attaching different blocks in the 3D scenario, and programming the interaction between visual blocks and sprites in the cabinet arcade game.

3 Conclusions and Related Work

The porting of the 3D game to the VR setting has been made possible by adopting a kit of Meta Oculus Quest 2 HMDs using OpenVR to have a project compatible with all Virtual Reality platforms. Moving within the VR game scenario was made possible through the teleport functionality and the arrangement of the various game areas to teleport the player, unlike the 3D version, where the booth controls were purely aesthetic. In the VR version, the player must directly use the pads and buttons of the cabinets to interact with the game inside them. It was also necessary to transfer the commands to the gamepad. The UEBlockly screen is opened by pressing a key. Each of its elements can be selected via two raycasts generated by the virtual hands. A block can be grabbed via a dedicated command and moved to the editor area. Similarly, a virtual keyboard can be activated via a special command and its keys pressed via the raycasts as shown in Fig. 4.

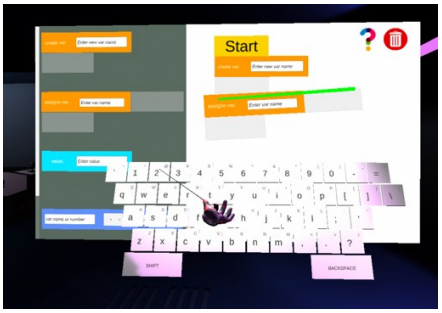

Fig. 4. The UE Blockly editor and the virtual keyboard in VR.

Several approaches for enriching learning environments and applications via VR technologies exist, see e.g. [6, 11]. Mozilla Hubs is a virtual collaboration platform for creating 3D scenarios that can be run directly on a web browser. Mozilla Hubs has been recently used for remote teaching, e.g., see [12]. Mozilla Hubs does not provide programmable virtual objects if not for basic functionalities.

Concerning 3D game development tools, we mention CoSpaces and Unreal. CoSpaces combines 3D VR/AR and coding frameworks very interestingly. Indeed, it provides a 2D/3D scene editor integrated with a block programming editor used to write scripts to animate game objects. The block editor provides full-fledged programming inspired by Google Blockly, visual languages enriched with blocks for 3D animation. Unreal Engine is a more sophisticated 3D game engine. Regarding the programming language, Unreal uses C++, while Unity uses C#, a language easier to use and learn. Unreal also provides a visual language called Blueprints to simplify game development. Concerning the Virtual Reality engine, both tools adopt the SteamVR package, which offers a set of libraries to create and test Virtual and Augmented Reality applications for the most common HMDs.

It is important to remark that CoSpaces, Unity, and Unreal are 3D development tools not designed for an immersive coding experience. At least in the proposed interaction mode of the three tools, players have no more access to the code/block editor when immersed in the game. Therefore, they modify the behavior of virtual objects if not according to the rules defined a priori by the designer.

Our proposal extends players' experience by providing a virtual block editor for a fully immersive coding experience. Arcade games are particularly suitable for education [13], in general, and coding exercises because of their simplicity (the Code.org web platform is an example). In our setting, the use of Virtual Reality allowed us to provide a richer experience to coding beginners, to easily add, modify, or delete exercises within a given cabinet, and easily adapt the proposed activity to the age of the participants.

The learning outcome of our experiment was to create new ways of stimulating curiosity towards computer science and, specifically, programming showing several different ways to interact and control a computer application. The possibility of using a coding editor such as UEBlockly during an immersive experience opens the way to a new class of coding tools in which players can interact with the virtual scenario to compose scripts to continue the game in a spirit like escape rooms and other types of exploratory VR games.

The preliminary work we presented needs further improvements and assessment regarding the human-computer interface and usability and a comprehensive analysis of learning outcomes. Indeed, the effect of immersive interfaces and environments on Computer Science teaching is of potentially great interest to the community [8]. From the development point of view, interaction with the coding blocks and the text input (to write variable names and assign them values) should be better analyzed, see [14]. Moreover, usability analysis with subjects of different ages, skills, and expertise in VR and coding, are part of our current and future work.

References

1. Haas, J. K.: A history of the unity game engine (2014).
2. Scavarelli, A., Arya, A., Teather, R.J.: Virtual reality and augmented reality in social learning spaces: a literature review. Virtual Reality 25, 257–277 (2021).
3. Wing, J.: Computational Thinking, Communications of the ACM, 49 (3), 33–35 (2006).
4. Denning, P. J., Tedre, M.: Computational thinking. Cambridge (2019).
5. Nardelli, E.: Do we really need Computational Thinking? Communications of the ACM. 62 (2): 32–35 (2019).
6. Pirker, J., Holly, M., Dengel, A., Safikhani, S.: Virtual Reality in Computer Science Education: A Systematic Review. Research Gate. In Proc. VRST'20: 26th ACM Symposium on Virtual Reality Software and Technology, (2020).
7. Mikropoulos, T. A., Natsis, A.: Educational virtual environments: A ten-year review of empirical research. (1999–2009).

8. Dengel, A. How Important is Immersion for Learning in Computer Science Replugged Games? In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE), pp. 1165-1171, 2020
9. Dengel, A.What Is Immersive Learning?. In 2022 8th International Conference of the Immersive Learning Research Network (iLRN), pp. 1-5 (2022), Austria (2022)
10. Horst, R., Naraghi-Taghi-Off, R., Diez, S., Uhmann, T., Müller, A., Dörner, R , Funplogs–a serious puzzle mini-game for learning fundamental programming principles using visual scripting. Advances in Visual Computing: 14th International Symposium on Visual Computing (ISVC), USA (2019)
11. Pirker, J., Dengel, A.: The potential of 360 virtual reality videos and real VR for education—a literature review. IEEE computer graphics and applications, 41(4), pp. 76-89 (2021)
12. Eriksson, T.: Failure and Success in Using Mozilla Hubs for Online Teaching in a Movie Production Course. 7th International Conference of the Immersive Learning Research Network (iLRN), pp. 1–8, Eureka, CA, USA (2021).
13. Schell, J.: The Art of Game Design: A Book of Lenses, CRC Press (2008).
14. Dudley, J., Benko, H., Wigdor, D., Kristensson.: P. O. Performance envelopes of virtual keyboard text input strategies in virtual reality. In IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 289-300 (2019)