The Visual Computer
https://doi.org/10.1007/s00371-020-01919-0

ORIGINAL ARTICLE

Immersive visual scripting based on VR software design patterns for experiential training

Paul Zikas[1] · George Papagiannakis[2] · Nick Lydatakis[1] · Steve Kateros[1] · Stavroula Ntoa[3] · Ilia Adami[3] · Constantine Stephanidis[4]

Abstract
Virtual reality (VR) has re-emerged as a low-cost, highly accessible consumer product, and training on simulators is rapidly becoming standard in many industrial sectors. However, the available systems are either focusing on gaming context, featuring limited capabilities or they support only content creation of virtual environments without any rapid prototyping and modification. In this project, we propose a code-free, visual scripting platform to replicate gamified training scenarios through rapid prototyping and VR software design patterns. We implemented and compared two authoring tools: a) visual scripting and b) VR editor for the rapid reconstruction of VR training scenarios. Our visual scripting module is capable of generating training applications utilizing a node-based scripting system, whereas the VR editor gives user/developer the ability to customize and populate new VR training scenarios directly from the virtual environment. We also introduce action prototypes, a new software design pattern suitable to replicate behavioral tasks for VR experiences. In addition, we present the training scenegraph architecture as the main model to represent training scenarios on a modular, dynamic and highly adaptive acyclic graph based on a structured educational curriculum. Finally, a user-based evaluation of the proposed solution indicated that users—regardless of their programming expertise—can effectively use the tools to create and modify training scenarios in VR.

Keywords Virtual reality · Authoring tool · VR training · Visual scripting

1 Introduction

Virtual reality has advanced rapidly, offering highly interactive experiences, arousing interest in both the academic and the industrial community. VR is characterized by highly immersive and interactive digital environments where user experiences another dimension of possibilities. As already known from conducted trials [2,17], the training capabilities of VR simulations offer skill transfer from the VR to real-life proposing an effective tool to fit in modern curricula. From pilots to surgeons, VR has a strong impact on training due to embodied cognition, psychomotor capabilities (dexterous use of hands) and high retention level [12].

✉ Paul Zikas
paul@oramavr.com

George Papagiannakis
george.papagiannakis@oramavr.com

Nick Lydatakis
nick@oramavr.com

Steve Kateros
steve@oramavr.com

Stavroula Ntoa
stant@ics.forth.gr

Ilia Adami
iadami@ics.forth.gr

Constantine Stephanidis
cs@ics.forth.gr

1 ORamaVR, Heraklion, Greece

2 ORamaVR, Institute of Computer Science, Foundation for Research and Technology-Hellas, Department of Computer Science, University of Crete, Heraklion, Greece

3 Institute of Computer Science Foundation for Research and Technology-Hellas, Heraklion, Greece

4 Institute of Computer Science, Foundation for Research and Technology-Hellas, Department of Computer Science, University of Crete, Heraklion, Greece

Published online: 08 August 2020

Authoring tools encapsulate key software functionalities and features for content creation. The software architecture of such system empowers programmers with the necessary tools for content creation. However, existing platforms do not sufficiently propose a complete methodology to reconstruct a training scenario in virtual environments. Training simulations are often implemented in modern game engines using native tools without any customization or specially designed features for generating of immersive scenarios rapidly. In addition, there are no prototyped software patterns specially formulated for VR experiences, leading to complex implementations and lack of code reusability.

Previously, we have proven that our VR training platform [3] makes medical training more efficient. In a revolutionary clinical study [2] in cooperation with New York University that established—for the first time in the medical bibliography—skill transfer and skill generalization from VR to the real Operating Room in a quantifiable, measurable ROI.

In this project, we propose a visual scripting system capable of generating VR training scenarios following a modular Rapid Prototyping architecture. Inspired from game programming patterns, we implemented new software design patterns named *Actions* for VR experiences to support a variety of commonly used interactions and procedures within training scenarios offering great flexibility in the development of immersive VR metaphors. We designed our solution as a collection of authoring tools combining a visual scripting system and an embedded VR editor forming a bridge from product conceptualization to product realization and development in a reasonably fast manner without the fuss of complex programming and fixtures. Our goals and design decisions were the following:

- **Educational pipeline:** We are interested in representing an educational process into an efficient data structure, for simple creation, easy maintenance and fast traversal.
- **Modular architecture:** To support a wide variety of interactions and different behaviors within the virtual environment, we want our system to integrate a modular architecture of different components linked into a common structure.
- **Code-free SDK:** Our intentions were to develop a platform where users can create VR training scenarios without advanced programming knowledge. We also aim to study techniques for the visual creation of VR experiences.
- **Rapid prototyping:** We are interested in making reusable prototyped modules to implement more complex interactive behaviors derived from elementary blocks.
- **VR software design patterns:** We aim to support a large number of interactive behaviors in VR applications to promote new software patterns specially formulated to speed up content creation in VR.

This paper is organized as follows. In Sect. 2, we present the state-of-the-art in training simulations and similar authoring tools. In Sect. 3, we introduce our solution with a brief description of our software modules. Section 4 presents the training scenegraph architecture. Section 5 describes the Action Prototypes and our rapid prototyping solution. Section 6 presents the visual scripting tool and Section 7 the VR editor. In Section 8, we discuss our results from the evaluation process. Section 9 concludes and defines the future work.

2 Related work

In this section, we present the state-of-the-art in VR training, its impact on education and similar authoring platforms.

2.1 The impact of VR in training and education

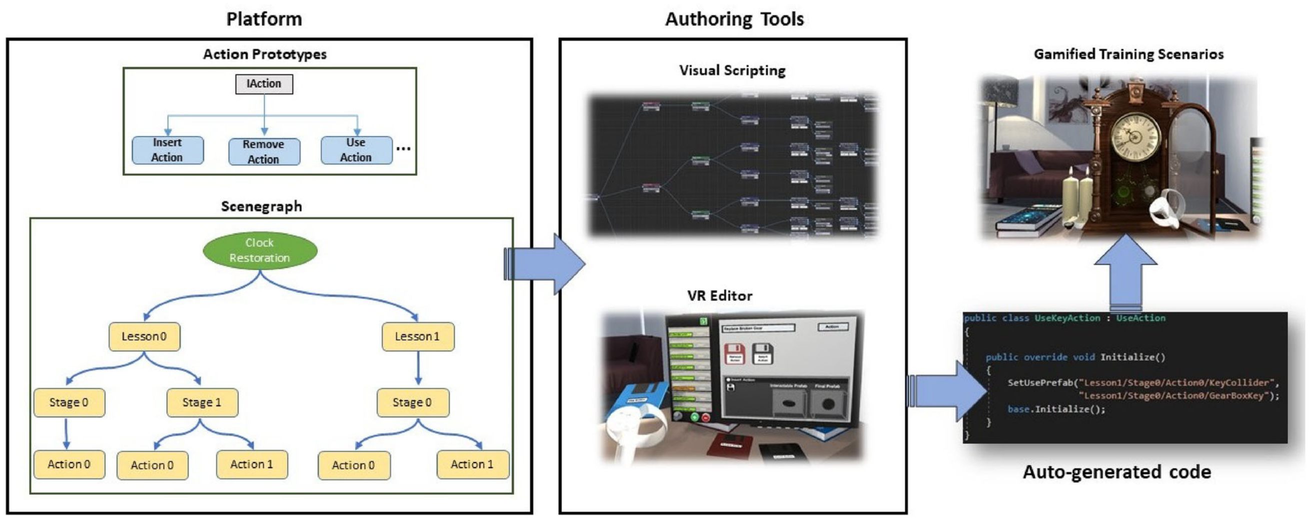
The engagement of education with novel technological solutions provides new opportunities to increase collaboration and interaction through participants, making the learning process more active, effective, meaningful and motivating [5]. Collaborative VR applications for learning [13], studies for the impact of VR in exposure treatment [7] as well as surveys for human social interaction [20] have shown the potential of VR as a training tool. The cognitive aspect of VR learning is already known from conducted trials [11]. Recent examples are featuring the learning capabilities of VR in surgical simulations [21] with remarkable results.

Focusing on the educational factor, the use of VR for knowledge transfer and e-learning is now extended as the R&D grows around entire VR environments where the learning takes place [16]. Virtual Reality rapidly increases its potential and influence on e-learning applications [10] by taking advantage of two basic principles: a) the embodiment [25] and b) the increased knowledge retention [6] with immersive environments capable of presenting a realistic scenario as it is, as it would be in real-life.

2.2 Authoring tools for content creation

The main concept behind authoring tools is to develop frameworks capable of generating content with minimal changes to speed up content creation while improving product maintenance.

BricklAyeR [26] proposes a collaborative platform designed for users with limited programming skills that allows the creation of Intelligent Environments through a building-block interface. Another interesting project is ARTIST [15], a plat-



```
public class UseKeyAction : UseAction
{

    public override void Initialize()
    {
        SetUsePrefab("Lesson1/Stage0/Action0/KeyCollider",
                     "Lesson1/Stage0/Action0/GearBoxKey");
        base.Initialize();
    }
}
```

Fig. 1 The architectural diagram of our system. The platform consists of a training scenegraph along with the action prototypes. In a higher hierarchy, the authoring tools (visual scripting and VR editor) are facilitating tools to generate interactive behaviors in the virtual environment. Finally, the training scenarios are implemented from the auto-generated code

form that provides tools for real-time interaction between human and non-human characters to generate reusable, low cost and optimized MR experiences. Its aim is to develop a code-free system for the deployment and implementation of MR content while using semantically data from heterogeneous resources.

Another authoring tool, ExProtoVAR [23], generates interactive experiences in AR featuring development tools specially designed for non-programmers, without necessarily a technical background with AR interfaces. In the field of interactive storytelling, StoryTec [14] platform facilitates an authoring tool to generate and represent storytelling-based scenarios in various domains (serious games, e-learning and training simulations).

2.3 Visual programming

Visual programming is getting more publicity as more platforms and tools are emerging. We can separate them into two categories according to their visual appearance and basic functionalities: a) block-based and b) node-based scripting languages

Block visual languages consist of modular blocks that represent fundamental programming utilities. OpenBlocks [24] proposes an extendable framework that enables application developers to build a custom block programming system by specifying a single XML file. Google's online visual scripting platform Blocky [22] uses interlocking, graphical blocks to represent code concepts like variables, logical expressions, loops and other basic programming patterns to export blocks to programming languages like JavaScript, Python, PHP and Lua.

On the other hand, node-based visual languages represent structures and data flow using logical nodes to reflect a visual overview of data flow. GRaIL [9] was one of the first systems that featured a visual scripting method for the creation of computer instructions based on cognitive visual patterns. More recently, Unity3D game engine has recently announced at their 2020 roadmap [4] an embedded node-based editor and a visual scripting system that will launch with their next update.

2.4 Editing directly from the VR environment

The development of authoring tools in virtual reality systems led to the integration of sophisticated functionalities. One of them is the implementation of immersive VR editors for the reconstruction of digital worlds directly from within the virtual environment.

In SIGGRAPH 2017, Unity technologies presented EditorVR [8], an experimental scene editor that encapsulates all the Unity's features within the virtual environment giving developers the ability to create a 3D scene while wearing the VR headset. EditorVR supports features for initially laying out a scene in VR, making adjustments to components and building custom tools. Except from game engines, model editors are also emerging into immersive VR model editing. MARUI [1] is a plugin for Autodesk Maya that lets designers perform modeling and animation tasks within the virtual environment.

The available VR editors feature scene management capabilities with intuitive ways to build a scene directly from within the virtual environment. However, there are no available authoring tools to offer a complete system for developing a behavioral VR experience including both the design and the programming aspect. The mentioned VR editors are mostly focused to showcase examples of interaction and scene creation.

The state-of-the-art shows that the mentioned VR platforms do not provide sufficient tools to generate training simulations nor a complete methodology for representing an educational process in VR. There is no similar platform which encapsulates functionalities for training using visual scripting and VR Editor combined. In addition, the bibliography shows that the available visual scripting platforms lack of solutions for VR training as they mostly concentrated on the visualization of simple systems and learning of programming.

For rapid adaptation to variations, a schematic representation of VR experiences is critical to replicate training scenarios and create a customized platform able to generate new content with minimal changes. Such system is not currently available and as a result developers have to implement content from scratch using third-party modalities and integrating various SDKs into a single project. This approach will lead to complex projects or even compatibility issues between the different modalities. It is evident that a unified platform is required to manage the communication of its embedded modules while offering authoring tools for rapid content creation without external modules required.

3 Our solution

The main goal of this project is to implement and compare three different authoring mechanics a) prototyped scripting, b) visual scripting and c) VR editor for rapid reconstruction of VR training scenarios based on our newly defined VR software design patterns. In other words, the proposed system facilitates a VR playground to recreate training scenarios. From the developer's perspective, this system constitutes a Software Development Kit (SDK) to generate VR content, which follows a well-structured educational pipeline. Our platform was built as a plugin for Unity3D engine for effective setup and distribution.

We introduce the following contributions:

- **Training scenegraph:** We developed a dynamic, modular tree data structure to represent the training scenario following a well defined educational curriculum. A training scenegraph tree stores data regarding the tasks where the trainee is asked to accomplish, dismantling the educational pipeline into simplified elements and focusing on one step at the time.
- **Action prototypes:** We designed reusable prototypes based on VR software design patterns to transfer behaviors from the real to the virtual world. Action prototypes populate the training scenegraph with interactive tasks for the user to accomplish. They introduce a novel methodology specially formulated for the development of interactive VR content.
- **Visual scripting:** We integrated a Visual Scripting system as an authoring tool to export training scenarios from a node-based, coding-free user interface.
- **VR editor:** We embedded a run-time VR Editor within the training scenarios to give user the ability to customize and create new scenarios directly from within the virtual environment.
- **Pilot applications:** Utilizing the proposed system, we generated two pilot training scenarios: a) a REBOA (Resuscitative Endovascular Balloon Occlusion of the Aorta) training scenario and b) an antique clock restoration.

In the following sections, we present the software modules and key functionalities of our system.

4 The training scenegraph

To achieve a goal, whether it is the restoration of a statue, the repair of an engine's gearbox or a surgical procedure the trainee needs to follow a list of tasks. We are referring to those tasks as *Actions*.

A simple visualization of a training scenario containing Actions would be to link them in a single line one after another. However, in complex training scenarios, a sequential representation would not be very convenient due to the absence of classification and hierarchical visual representation. For this reason, we implemented the training scenegraph architecture. Training scenegraph is a tree data structure representing the tasks/Actions of a training scenario. The root of the tree holds the structure, on the first depth we initialize the *Lesson* nodes, then the *Stage* nodes and finally at leaf level the *Action* nodes.

One of the main principles of this project was to modify the training scenegraph and Actions using three different editors (scripting, visual scripting and the VR editor). To achieve this, the scenegraph data are stored to an xml file offering easy maintenance.

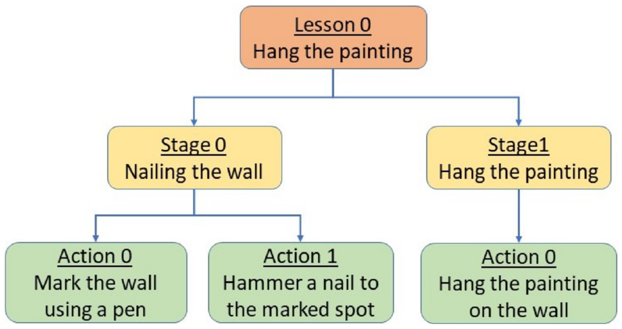


Fig. 2 An example of a training scenegraph tree representing the simple scenario of hanging a paint on the wall

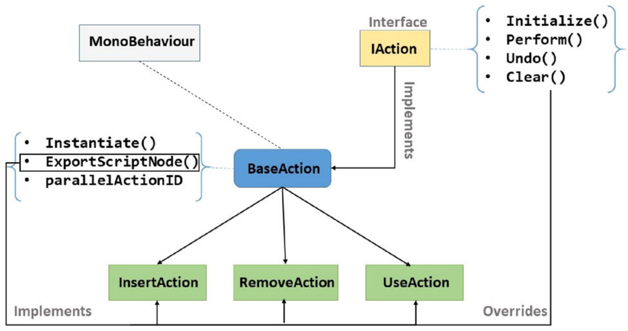


Fig. 3 Action Prototypes Architecture diagram

5 Action prototypes

In this section, we analyze how we implemented new VR software design patterns thought rapid prototyping.

5.1 The IAction interface

The Action object reflects a flexible structural module, capable of generating complex behaviors from basic elements. This also reflects the concept idea behind the training scenegraph; provide developers with fundamental elements and tools to implement scenarios from basic principles. Each Action is described by a script containing its behavior in means of physical actions in the virtual environment. Technically, each Action script implements the IAction interface, which defines the basic rules every Action should follow, ensuring that all Actions will have the same methods and structure. Below, we present the components of IAction interface.

– **Initialize:** This method is responsible to instantiate all the necessary 3D objects for the Action to operate.
– **Perform:** This method completes the current Action and deletes unused assets before the next Action starts.
– **Undo:** This method resets an Action including the deletion of instantiated 3D assets and the necessary procedures to set the previous Actions.
– **Clear:** Clears the scene from initialized objects and references from the Action.

Designing a shared interface among the structural elements of a system is the first step to prototype commonly used components. This methodology is both beneficial for the user and the developer: a) users are introduced with interactive patterns that are familiar with, avoiding complex behaviors while b) developers are following the same implementation patterns.

5.2 From actions to VR design patterns

To make our system more efficient, we have to limit the capabilities of the Action entity targeting simple but commonly used tasks in training. Modeling those behaviors, we will generate a pool of generic behavioral patterns suitable for VR applications.

The implementation of Action prototypes was highly inspired by Game Programming Patterns [19] as an alternative paradigm for VR experiences. The immersion of virtual environments causes the implementation of programming patterns to fit into a more interactive way of thinking. For this reason, the software patterns developed in this project designed to match the needs for interactivity, embodied cognition and physicality on VR experiences. For this reason, we implemented the following Action Prototypes:

– **Insert action:** is referring to the insertion of an object to a predefined position. Technically, to implement an Insert Action, the developer needs to set the initial and the final position of an object.
– **Remove action:** describes a step in which the user has to remove an object using his hands. To implement a Remove Action the developer needs to define the position where the object will be instantiated, for user to reach and remove it.
– **Use action:** refers to a step where the user needs to interact with an object over a predefined area for a period of time. Figure 3 illustrates an architectural diagram of Action Prototypes to visualize their dependencies.

Action Prototypes constitute a powerful software pattern to implement interactive tasks in VR experiences. Unitizing Action Prototypes, developers can replicate custom behaviors with a few lines of code taking advantage of their abstraction and reusability. New Action Prototypes can be easily implemented and extended due to their abstraction.

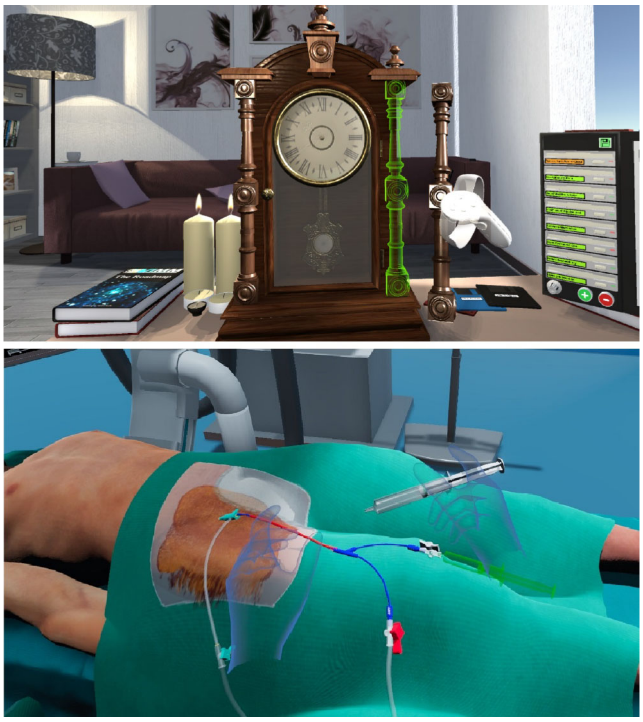


Fig. 4 Top: Insert Action from the clock's maintenance use case. Bottom: Insert Action from the REBOA use case. The green holograms indicate the correct position of 3D objects

5.3 Alternative paths

The Action prototypes propose a new design pattern for VR experiences, a modular building block to develop applications in combination with the training scenegraph. However, the proposed training scenegraph architecture generates VR experiences following a linear pipeline of Actions where user needs to complete a predefined list of tasks. In order to transform the training scenegraph from a linear tree into a dynamic graph, we introduced Alternative Paths.

A training scenario can lead to multiple paths according to the user's actions and decisions, scenegraph adapts. In addition, certain actions or even wrong estimations and technical errors may deviate the original training scenario from its normal path causing the training scenegraph to modify itself accordingly. Except for backtracking after wrong estimations and errors, the Alternative Path mechanic is also used in situations where the trainee needs to make a particular decision over a dilemma. From a technical perspective, Actions are able to trigger alternative path events. Those events will be advanced to training scenegraph informing about the necessary follow-up actions. The scenegraph tree will update its form accordingly by pruning or adding new nodes to its structure to adapt to the new circumstances.

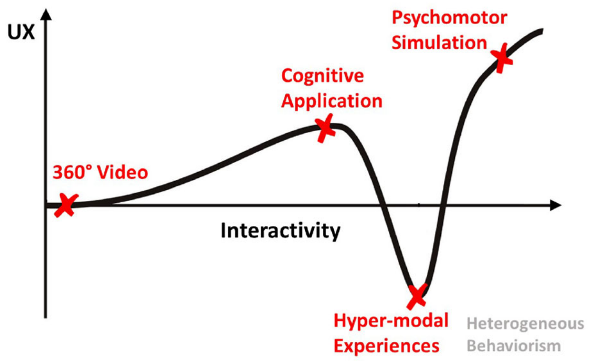


Fig. 5 The uncanny valley of interactivity. Correlation between User experience (UX) and interaction in VR

5.4 The uncanny valley of interactivity and VR UX

After experimenting with various design patterns and interaction techniques for VR, an interesting pattern appeared regarding the correlation of user experience and the interactivity of the VR application (Fig. 5). An immersive experience relies significantly on the implemented interactive capabilities that form the user experience. As a result, to make an application more attractive in means of UX a more advanced interactive system is needed. However, as we implement more complex interaction mechanics there is a point in timeline where the UX drops dramatically. At this point, the application is too advanced and complex for the user to understand and perform the tasks with ease. We characterize this feature as heterogeneous behaviorism meaning that user's actions do not follow a deterministic pattern resulting in the inability to complete the implemented Actions due to their incomprehensible complexity.

In contrast, applications with limited interactivity follow a linear increase in their user experience. From applications where users are only observers (360VR videos) to cognitive applications, the interaction curve is linear and VR experiences easy to understand.

There are two ways to overcome the uncanny valley: a) drastically enhance the interactivity capabilities aiming for a psycomotor simulation or b) reducing complexity aiming for a cognitive application and make users understand how they are supposed to act in the virtual world. Over-passing the valley of interactivity, applications are evolving rapidly to follow a psychomotor methodology integrating embodied cognition for better UX. In contrast, cognitive applications lack of realism, but they offer intuitive and easy to follow mechanics. The choice depends on the design and the genre of the application.

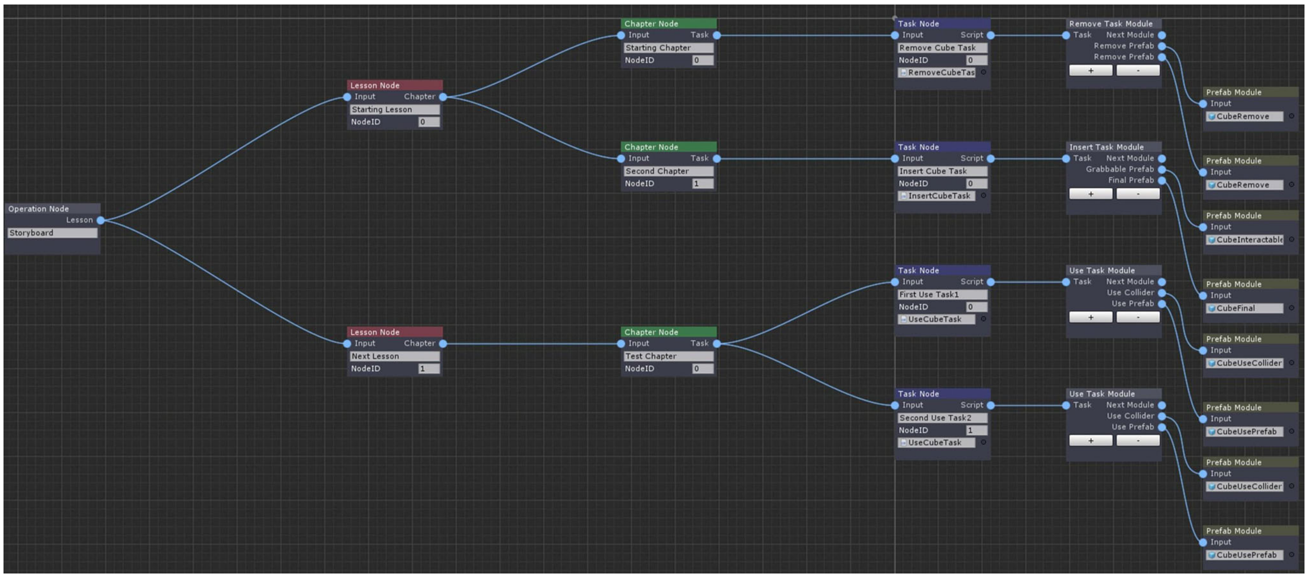


Fig. 6 A training scenario visualized from the Visual Scripting Editor featuring from right to left: Lessons (red), Stages (green), Actions (blue), Action Scripts (gray) and Prefab nodes (brown). Prefabs are representing 3D objects in the virtual environment. For example, an Insert Action contains two prefabs: the interactable item and its final position

6 Visual scripting

The training scenegraph model is capable of generating applications from reusable fundamental elements (Actions) supporting basic insert, remove and use behaviors in VR. However, what is the next step? What can be done to enhance the development process and speed up content creation? The complexity of scenegraph xml may cause difficulties visualizing the scenegraph nodes, especially for extended training scenarios. Another point is the programming skills required do develop such experiences. Using the proposed architecture could be challenging for inexperienced programmers

To eliminate the mentioned difficulties, we introduce visual scripting as an authoring tool to manage, maintain and develop VR experiences. Visual scripting encapsulates all the functionalities from the base model while offering high visualization capabilities.

6.1 The visual scripting metaphor

The development of a visual scripting system as an assistive tool aimed to visualize the VR training scenario in a convenient way, if possible fit everything into one window. The simplicity of this tool was carefully measured to provide tool used also from non-programmers. From the beginning of the project, one of the main design principles was to strategically abstract the software building blocks into basic elements. The main idea behind this abstraction was the improvement of the visual scripting and VR editor tools since fundamental elements construct a better visual representation than complex ones. To render the visual nodes, we exploited Unity Node Editor Base (UNEB), an open-source framework, which provides basic node rendering.

Moving into the visual scripting metaphor, the training scenegraph data structure forms a dynamic tree, visualizing the scenario into a node-based editor with nodes linked together forming logical segments. To construct visual nodes, our system retrieves data from the Action scripts through reflection and run-time compilation. An example of a complete diagram representing a training scenario is illustrated in Fig. 6. Developers can utilize visual scripting to generate training scenarios through interactive UIs transforming the VR content generation into a coding-free process.

6.2 Dynamic code generation

Visual scripting generates run-time simple Action scripts utilizing the information provided from the visual input. After completing the visual construction of an Action, the next step is to generate the Action script to save the implemented behavior in a C# code script.

To write C# code run-time, we used CodeDOM [28], a build-in tool for .NET Framework that enables run-time code generation and compilation. The abstraction of Action prototypes offers an elegant implementation to generate each script using a single virtual method. To finalize the Action script, except the Action Type (Insert, Remove or Use) we also need the interactive behavior. Action prototypes retrieve this information directly from the visual scripting editor through the linked nodes relative to the Action module.

6.3 Expanding auto-generated scripts

Visual scripting generates a basic Action script containing the Initialize method, the minimum requirement for an Action to run properly. However, there are cases where developers need to implement significantly complex Action behaviors to enhance use experience.

Prototyped Actions were developed using a particular software architecture capable of providing the fundamental facilities but also customize Actions according to the developer's preferences. The Perform method can be overridden directly from the Action script to extend the Action's capabilities. The same principle is applicable to all the other virtual methods defined in the IAction interface (Undo, Initialize, etc.). For additional modifications, the best practice is to edit directly the generated script and override the declared IAction methods. In this way, we maintain simple scripts but also provide custom implementations upon request.

7 VR editor

The visual scripting system enhanced the usability and effectiveness of the scenegraph system to generate gamified training scenarios through a coding-free platform. The impact on content creation was very strong due to the additional tools and features that introduced. However, visual scripting lacks on one specific and rather important feature: the ability to design on-the-go behaviors and scenarios directly within the virtual environment. This feature will improve design capabilities while offering an intuitive way to modify applications directly from the virtual environment.

The implementation of VR editor was designed as an authoring tool on top of the training scenegraph architecture, utilizing the developed features of our system. This interactive tool reduces the time needed to produce training scenarios due to the rapid in-game generation of training scenarios. In addition, certain interactive behaviors are better designed directly from VR due to the 3D perspective of the medium.

7.1 The VR metaphor

The main concept behind the implementation of our VR editor focuses on an interactive system with floppy disks and a personal computer. Figure 7 illustrates the design of our VR editor along with its various components and floppy disks. The training scenegraph nodes are represented by floppy drives on the left side of the screen. Action scripts are initialized as floppy disks, each one holds the script behavior that defines the 3D objects relative to the Action. There are three types of floppy disk separated with unique coloration; blue

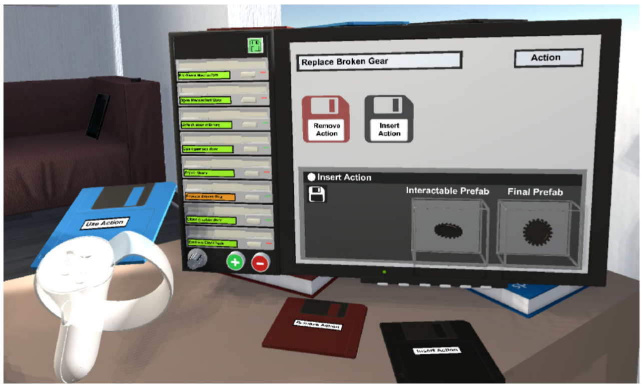


Fig. 7 Interacting with the VR editor. User holds a Use Action preparing to generate the Action behavior

disks represent Use Actions, red disk the Remove Actions and black disks the Insert Actions.

7.2 Generate actions and parametrization on-the-go

The functionality with the higher impact on the VR editor is by far the ability to modify and parametrize Actions on-the-go. This was also the main reason that led us to implement the VR editor as an additional authoring tool within the virtual environment, to support coding-free development and give the user the ability to modify or even generate new behaviors while experiencing the training scenario.

Users can customize the scenegraph through VR editor by adding or deleting Scenegraph nodes to match their needs. This functionality has a serious impact on users that want to parametrize existing VR training scenarios or create their own without having any programming knowledge. We provide this ability via an interactive UI on the VR editor with physical buttons where users can modify the training scenegraph.

The next step is the script generation from the VR editor. To generate a new Action, users need to insert a floppy disk into the drive representing the Action script (Insert, Remove or Use). The system will register the insertion of floppy disks and an empty Action script will appear on VR editor screen ready for modification.

With VR editor, users are no longer just observers, they can modify the training scenarios on-the-go, implement new ideas and fix wrong Action behaviors without specialized programming knowledge.

8 Evaluation

To examine the overall experience of using our system, we conducted a preliminary user-based evaluation with 18 users [18]. The main research questions were the following:

Table 1 Participants' demographics with regard to their expertise in VR and SD

	None	Little	Medium	Good	Excellent
VR	7	2	0	5	4
SD	8	1	0	2	7

– For the VR training application: what is the overall perceived quality of the VR training environment and perceived educational value?
– For the Visual scripting tool: can users successfully complete basic programming tasks and how do they rate the overall experience?
– For the VR editor tool: can users successfully complete basic adjustments to an existing training scenario and how do they rate the overall experience?

8.1 Methodology and participants

The experiment was divided into three separate sessions, one for each tool. In the first session, the participants were asked to restore an antique clock following the instructions given by the VR training application (Clock repair scenario). In the second session, the participants were shown the capabilities and functionalities of the Visual Scripting tool. Then, they were asked to use the tool to generate code for a "Use" action (*'Use the sponge to wipe dirty spot on the clock'*) and a "Remove" action (*'Remove seal from two-sided gear'*). Finally, in the third session, the participants were asked to complete two tasks to adjust the clock restoration training scenario directly from the VR environment using the VR editor tool.

A 10-point Likert Scale questionnaire was given at the end of each session to rate the parameters identified in the research questions. For the 'educational' value of the VR training application, participants were also asked to indicate which steps they retained regarding the restoration process. In addition, metrics such as the number of help requests and time on task were recorded for further analysis of the results. Finally, at the end of the experiment, a semi-structured interview was conducted to capture participants' general impression of the whole system.

Eighteen people participated in our experiment, 11 males and 7 females. All users were in the 25–35 age range. They were selected based on the level of expertise in using VR applications and level of expertise in Software Development (SD), ensuring an equal number of expert and non-expert participants in each one of the two categories, as illustrated in Table 1 below [18,27]

Table 2 VR training application - rating scores

	All	Experts	Non-experts
Percieved quality of VR experience			
Avg	8.67	8.22	9.11
StDev	1.029	1.202	0.601
CI	0.512	0.924	0.462
Percieved educational value			
Avg	8.78	8.67	8.89
StDev	8.88	1.00	0.78
CI	0.43	0.769	0.601
Recall activity score			
Avg	9.26	8.93	9.63
StDev	1.30	1.66	0.73
CI	0.65	1.28	0.56

CI 95% Confidence interval

8.2 Results

8.2.1 First session: VR training application

The perceived quality of the VR experience was on average highly rated by all participants (8.6/10). The average rating of VR experts was somewhat lower than that of non-experts, but the difference was not statistically significant, as revealed by a paired t-test analysis ($t(8) = -1.83$, $p = 0.1$).

Equally high was the overall average rating score the application received in terms of the perceived educational value (8.78/10). Small differences were exhibited between VR experts and non-experts; however, no statistical significance was identified ($t(8) = -0.45$, $p = 0.66$).

The participants also scored rather high in the exercise where they indicated which steps they could recall from the restoration process (9.26/10). A comparison of the achieved score between the two groups did not indicate a statistically important difference ($t(8) = -1.08$, $p = 0.31$).

Time on task and the number of help requests were recorded to support further analysis of participants' ratings with regard to the effort they invested. All participants were able to complete the steps of the training scenario successfully and within a reasonable time (2:50 minutes on average). Slightly higher completion time was recorded on average for the non-experts, but with no statistical significance ($t(8) = -1.5$, $p = 0.17$). However, there was a statistically significant difference in the number of help requests between the two groups ($t(8) = -4.6$, $p = 0.001$), as expected, due to the inexperience of the users.

8.2.2 Second session: visual Scripting tool

All participants rated highly the perceived easiness of completing the two given script tasks with the Visual Scripting

Table 3 VR training application—time on task and number of help requests

	All	Experts	Non-experts
Time (min)			
Avg	2:50	2:38	3:03
StDev	0:59	1:00	1:00
CI	0:29	0:46	0:46
# of help requests			
Avg	2.56	1.78	3.33
StDev	1.20	0.97	0.87
CI	0.60	0.75	0.67

Table 4 Visual Scripting—perceived task easiness for Task 1 ($T1$) and Task 2 ($T2$)

	All		Experts		Non-experts	
	$T1$	$T2$	$T1$	$T2$	$T1$	$T2$
Avg	8.00	8.28	8.44	8.67	7.56	7.89
StDev	1.19	0.95	1.13	1.00	1.13	0.78
CI	0.59	0.48	0.87	0.77	0.87	0.60

Table 5 Visual Scripting—time on task, number of help requests (for the entire scenario) and overall experience

	All	Experts	Non-experts
Time (min)			
Avg	12:36	9:41	17:31
StDev	4:14	1:02	1:32
CI	2:06	0:47	1:11
# of help requests			
Avg	2.89	2.11	3.67
StDev	1.41	1.54	0.71
CI	0.70	1.18	0.54
Overall experience			
Avg	8.61	7.88	8.44
StDev	0.98	0.78	1.13
CI	0.49	1.13	0.86

Table 6 VR Editor—perceived task easiness for Task 1 ($T1$) and Task 2 ($T2$)

	All		Experts		Non-experts	
	$T1$	$T2$	$T1$	$T2$	$T1$	$T2$
Avg	7.50	7.06	8.29	7.67	6.89	6.44
StDev	1.26	1.39	1.38	1.66	0.78	0.73
CI	0.67	0.69	1.28	1.27	0.60	0.56

tool. However, paired t-testing revealed a statistically significant difference in the scores received for Task 1 by the SD experts and by the non-experts; $t(4) = -5.66$, $p = 0.005$. Similarly, there was a statistically significant difference in the scores received for Task 2 by the SD experts and the non-experts; $t(8) = 2.8$, $p = 0.02$.

These observations are aligned with the differences in the measurements of time on task and number of help requests between the SD experts and the non-experts. As shown in Table 4, non-experts required both more time and assistance. Paired t-testing confirmed that the differences carried a statistical significance both for time on task ($t(8) = -14.69$, $p = 0.0000004$) and number of help requests ($t(8) = -2.25$, $p = 0.05$).

Nevertheless, it is interesting that the additional required effort by non-experts to complete the tasks did not affect their overall experience with the tool. In fact, all participants rated highly the overall experience of using this tool (8.61/10), without any statistical difference between the two groups; $t(8) = -1.1$, $p = 0.3$.

8.2.3 Third session: VR editor tool

The participants also rated highly the perceived easiness of completing the two tasks for the VR Editor tool evaluation. Just like in the Visual Scripting tool, the SD experts found the tasks easier than non-experts, a difference which was identified as statistically important both for Task 1 ($t(8) = 2.56$, $p = 0.02$) and Task 2 ($t(8) = 2.34$, $p = 0.04$).

Differences were exhibited on the time on task and the number of help requests between the SD experts and the non-experts as expected and in alignment with the perceived ease of completing the tasks. Paired t-testing confirmed that the differences carried a statistical significance both for time on task ($t(8) = -7.1$, $p = 0.0009$) and for the number of help requests: ($t(8) = -3.05$, $p = 0.01$).

Just like the observed results in the second session with regard to the overall experience, non-experts gave equally high rates to the overall experience as the experts, despite the extra time and effort required to complete the tasks. The overall experience score was 7.61/10, while no statistically important difference between the groups was observed ($t(8) = 0.5$, $p = 0.6$).

In conclusion, all participants regardless of their expertise in VR and SD were able to successfully complete the tasks. The non-experts did—as expected—require more time and assistance, but did not seem to affect their overall experience in using the tools. The results of the evaluation matched the general sentiment of the participants about the overall suite expressed through the positive comments in semi-structured interviews, as well as through responding to a corresponding question in the questionnaire (Fig.8).

Table 7 VR Editor—time on task, number of help requests (for the entire scenario) and overall experience

	All	Experts	Non-experts
Time (min)			
Avg	13:24	9:08	17:39
StDev	5:00	2:52	2:03
CI	2:29	2:12	1:35
# of help requests			
Avg	2.83	2.22	3.44
StDev	1.04	0.97	0.73
CI	0.52	0.75	0.56
Overall experience			
Avg	7.61	7.77	7.44
StDev	1.09	1.09	1.13
CI	0.54	0.84	0.86

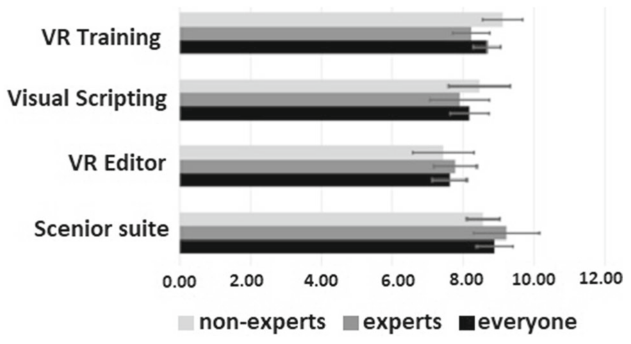


Fig. 8 Experience scores for each one of the authoring tools and the overall suite

9 Conclusions and future work

In this work, we presented a novel system capable of generating gamified training experiences exploiting its modular architecture and the authoring tools we developed. We introduced the scenegraph as a dynamic, acyclic data structure to represent any training scenario following an educational curriculum. In addition, we proposed a category of new software design patterns, the Action prototypes, specially formulated for interactive VR applications. Finally, we developed a visual scripting tool along with a VR editor to enhance the visualization and speed up content creation.

Our System has certain limitations linked with its components and functionalities. First of all, the evaluation process highlighted weaknesses in the interaction with the VR editor. Although it behaves well in Action customization, the script generation process is still complex due to the amount of information and steps needed from the user. In addition, some of its interactive components are not intuitive, resulting in the frustration of users when asked to implement certain behaviors. Finally, regarding the visual scripting editor, the real-time compilation process may cause performance issues in complex training scenarios and delay the initialization of scenegraph. These findings will be addressed, before proceeding to large-scale evaluations, to further evaluate the usability, usefulness and overall user experience of the tools.

To overcome the UX limitations, we aim to simplify our system by adding an additional observer on both the VR learning modules and the development pipeline. This observer will identify whenever the user does not know how to proceed and will provide with additional details in form of UIs and vocal guidance. In addition, we aim to include a holographic guidance during the VR training scenarios to enhance all Actions with visual information on how to complete each step.

The purpose of the preliminary evaluation was to get an overall impression of the authoring tools. This did not allow for in-depth analysis of each tool separately, in terms of effectiveness and efficiency. This is a known limitation that will be rectified by conducting further testing for each tool separately in future iterations, involving representative users from target user populations, such as the medical domain.

In the future, we aim to utilize computer vision to capture the trainer's movements from external cameras or directly from within the virtual environment to automatically generate interactive behaviors in VR. Another idea is to collect this data through video from a real-life scenario by monitoring the trainer and afterward processing the data using machine learning to extract important key features and construct a template of the training scenario.

Acknowledgements This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871793 (ACCORDION) and No727585 (STARS-PCP) and supported by Greek national funds (projects VRADA and vipGPU).

References

1. MARUI 3 plugin for Autodesk Maya. https://www.marui-plugin.com/marui3/. Accessed 30 July 2020
2. Our clinical trial (citation not provided for the reviewing process)
3. Our VR training platform (citation not provided for the reviewing process)
4. Unity 2020 roadmap. https://unity3d.com/unity/roadmap
5. Alsumait, A., Almusawi, Z.S.: Creative and innovative e-learning using interactive storytelling. Int. J. Pervasive Comput. Commun. **9**(3), 209–226 (2013). https://doi.org/10.1108/IJPCC-07-2013-0016
6. Andersen, S.A.W., Konge, L., Cayé-Thomasen, P., Sørensen, M.S.: Retention of mastoidectomy skills after virtual reality simulation training. JAMA Otolaryngol. Head Neck Surg. **142**(7), 635–640 (2016). https://doi.org/10.1001/jamaoto.2016.0454
7. Bouchard, S., Dumoulin, S., Robillard, G., Guitard, T., Klinger, E., Forget, H., Loranger, C., Xavier Roucaut, F.: Virtual reality compared with in vivo exposure in the treatment of social anxiety

disorder: a three-arm randomised controlled trial. Br psychiatr J Ment sci **210**, (2016). https://doi.org/10.1192/bjp.bp.116.184234
8. Ebrahimi, A., West, T., Schoen, M., Urquidi, D.: Unity: Editorvr. In: ACM SIGGRAPH 2017 Real Time Live!, SIGGRAPH '17, pp. 27–27. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3098333.3098918
9. Ellis, T.O., Heafner, J.F., Sibley, W.L.: The grail project: an experiment in man-machine communications (RM-5999-ARPA). RAND Corporation, Santa Monica (1969)
10. de Faria, J.W.V., Teixeira, M.J., de Moura Sousa Júnior, L., Otoch, J.P., Figueiredo, E.G, : Virtual and stereoscopic anatomy: when virtual reality meets medical education. J. Neurosurg. JNS **125**(5), 1105–1111 (2016)
11. Ganier, F., Hoareau, C., Tisseau, J.: Evaluation of procedural learning transfer from a virtual environment to a real situation: a case study on tank maintenance training. Ergonomics **10**(1080/00140139), 899628 (2014)
12. Greenleaf, W.: How vr technology will transform healthcare. In: ACM SIGGRAPH 2016 VR Village, pp. 1–2 (2016). https://doi.org/10.1145/2929490.2956569
13. Greenwald, S., Kulik, A., Kunert, A., Beck, S., Froehlich, B., Cobb, S., Parsons, S., Newbutt, N., Gouveia, C., Cook, C., Snyder, A., Payne, S., Holland, J., Buessing, S., Fields, G., Corning, W., Lee, V., Xia, L., Maes, P.: Technology and applications for collaborative learning in virtual reality. In: CSCL (2017)
14. Göbel, S., Salvatore, L., Konrad, R.: Storytec: A digital storytelling platform for the authoring and experiencing of interactive and non-linear stories. In: 2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution, pp. 103–110 (2008). https://doi.org/10.1109/AXMEDIS.2008.45
15. Kotis, K.I.: Artist–a real-time low-effort multi-entity interaction system for creating reusable and optimized MR experiences. Res. Ideas Outcomes **5**, e36464 (2019). https://doi.org/10.3897/rio.5.e36464
16. Monahan, T., McArdle, G., Bertolotto, M.: Virtual reality for collaborative e-learning. Comp. Educ. **50**(4), 1339–1353 (2008). https://doi.org/10.1016/j.compedu.2006.12.008
17. Murcia-López, M., Steed, A.: A comparison of virtual and physical training transfer of bimanual assembly tasks. IEEE Trans. Vis. Comp. Gr. **24**(4), 1574–1583 (2018). https://doi.org/10.1109/TVCG.2018.2793638
18. Nielsen, J.: Usability testing. In: Nielsen, J. (ed.) Usability Engineering, pp. 165–206. Morgan Kaufmann, San Diego (1993)
19. Nystrom, R.: Game Programming Patterns, 3rd edn. Genever Benning (2014)
20. Pan, X., Hamilton, A.: Why and how to use virtual reality to study human social interaction: the challenges of exploring a new research landscape. Br. J. Psychol. (2018). https://doi.org/10.1111/bjop.12290
21. Papagiannakis, P., Trahanias, G., Kenanidis, E., Tsiridis, E.: Psychomotor Surgical Training in Virtual Reality.Master Case Series and Techniques, pp. 827–830. Adult Hip, Springer, Cham (2017)
22. Pasternak, E., Fenicheland, R., Marshall, A.N.: Tips for creating a block language with blockly. In: 2017 IEEE Blocks and Beyond Workshop (B B), pp. 21–24 (2017). https://doi.org/10.1109/BLOCKS.2017.8120404
23. Pfeiffer-Leßmann, N., Pfeiffer, T.: Exprotovar: a lightweight tool for experience-focused prototyping of augmented reality applications using virtual reality. In: Stephanidis, C. (ed.) HCI International 2018—Posters' Extended Abstracts, pp. 311–318. Springer International Publishing, Cham (2018)
24. Roque, R.: Openblocks : an extendable framework for graphical block programming systems (2008)
25. Slater, M.: Implicit Learning Through Embodiment in Immersive Virtual Reality, pp. 19–33. Springer, Singapore (2017)
26. Stefanidi, E., Arampatzis, D., Leonidis, A., Papagiannakis, G.: BricklAyeR: A Platform for Building Rules for AmI Environments in AR, pp. 417–423 (2019)
27. Tullis, T., Albert, W.: Measuring the User Experience, Second Edition: Collecting, Analyzing, and Presenting Usability Metrics, 2nd edn. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2013)
28. Villela, R.: Working with the CodeDOM, pp. 155–177. Apress, Berkeley, CA (2019)

Paul Zikas is a software engineer with experience in AR and VR applications. He worked as an undergraduate researcher at the Computer Vision and Robotics Laboratory at FORTH studying Serious Games and Storytelling in Mixed Reality while participating in EU programs. As an enthusiast in Computer Graphics and Game Programming, he designed various Mixed Reality applications for educational and entertainment purposes in mobile and desktop. He earned his BSc and MSc in Computer Science from the University of Crete specializing in computer graphics. Currently, he is Lead Developer at ORamaVR and a PhD student at the University of Crete.

George Papagiannakis is a computer scientist specialized in computer graphics and virtual-augmented reality. He obtained his PhD in Computer Science at the University of Geneva in Switzerland in 2006, his M.Sc. in Advanced Computing at the University of Bristol and his B.Eng. in Computer Systems Engineering, at the University of Manchester. He is Associate Professor of Computer Graphics at the Computer Science department of the University of Crete, Greece and Affiliated Research Fellow at the Computer Vision and Robotics Laboratory in the Institute of Computer Science of the Foundation for Research and Technology Hellas and co-founder/CTO at ORamaVR.

Nick Lydatakis graduated from Computer Science department, University of Crete. He started as a research fellow at the Institute of Computer Science of the Foundation for Research and Technology - Hellas, on Computational Vision and Robotics Laboratory. He works in the fields of VR from 2015. He is currently working at ORamaVR as Head of Product Development.

Steve Kateros has obtained his bachelor degree at the Computer Science Department, University of Crete. A skilled C++, C# developer mostly focused in Game and Graphics Development and with a lot of interest in VR applications. A two year undergrad researcher at ICS-FORTH, and he is currently working at ORamaVR as a Product Manager and VR User Experience and Level Design.

Stavroula Ntoa holds a Ph.D. in “Information Systems and Human-Computer Interaction”, from the Computer Science Department of the University of Crete. She is a member of the Human-Computer Interaction (HCI) Laboratory of ICS-FORTH since 2000. She is experienced in the design, development and evaluation of accessibility software for motor-impaired users, and accessible web applications. She has expertise in UX design and evaluation in a number of projects in various contexts and application domains, including responsive web, big data, mobile, as well augmented and virtual reality applications. Her current work focuses on user experience design and evaluation in intelligent environments.

Illa Adami holds an M.B.A in Information Management from California State University, San Bernardino. She worked in the USA for seven years at Environmental Systems Research Institute (ESRI-California). Her main responsibilities included user requirements analysis, information architecture, and prototype designing for various web applications for the company. Ilia has been a member of the Human-Computer Interaction (HCI) Laboratory of the ICS - FORTH since 2007. She specializes in UX design, usability and accessibility evaluations in various technology domains and contexts and has participated in numerous scientific projects.

Constantine Stephanidis is Professor of Human Computer Interaction at the Department of Computer Science of the University of Crete. He is the Founder and Head (since 1989) of the Human - Computer Interaction Laboratory, and (since 2004) the Founder and Head of the Ambient Intelligence Programme at the Institute of Computer Science of FORTH, where he also served as Director between 2004 and 2016. He is the Founder and Editor-in-Chief (since 2000) of the Springer international journal “Universal Access in the Information Society” and the co-Editor (since 2016) of the T&F International Journal of Human Computer Interaction. http://www.ics.forth.gr/hci/Stephanidis.php