information

Article

TrainAR: An Open-Source Visual Scripting-Based Authoring Tool for Procedural Mobile Augmented Reality Trainings

Jonas Blattgerste *, Jan Behrends and Thies Pfeiffer

Faculty of Technology, University of Applied Sciences Emden/Leer, Constantiaplatz 4, 26723 Emden, Germany; thies.pfeiffer@hs-emden-leer.de (T.P.)
* Correspondence: jonas.blattgerste@hs-emden-leer.de

Abstract: Mobile Augmented Reality (AR) is a promising technology for educational purposes. It allows for interactive, engaging, and spatially independent learning. While the didactic benefits of AR have been well studied in recent years and commodity smartphones already come with AR capabilities, concepts and tools for a scalable deployment of AR are still missing. The proposed solution TrainAR combines an interaction concept, a didactic framework and an authoring tool for procedural AR training applications for smartphones. The contribution of this paper is the open-source visual scripting-based authoring tool of TrainAR in the form of a Unity Editor extension. With this approach, TrainAR allows non-programmer domain experts to create ("author") their own procedural AR trainings by offering a customized editor, while at any time programmers may decide to utilize Unity's full capabilities. Furthermore, utility and usability evaluations of several already developed TrainAR trainings (combined n = 317) show that TrainAR trainings provide utility in several contexts and are usable by the target groups. A systematic usability evaluation of the TrainAR Authoring Tool (n = 30) shows that it would be usable by non-programmer domain experts, though the learning curve depends on the media competency of the authors.

Keywords: augmented reality; handheld; authoring; content creation; 3D scanning; procedural training; scalable; education; learning; training

Citation: Blattgerste, J.; Behrends, J.; Pfeiffer, T. TrainAR: An Open Source Visual Scripting-Based Authoring Tool for Procedural Handheld Augmented Reality Trainings. *Information* **2023**, *14*, 219. https://doi.org/10.3390/info14040219

Academic Editors: Ramon Fabregat, Jorge Luis Bacca Acosta and N.D. Duque-Mendez

Received: 6 March 2023
Revised: 27 March 2023
Accepted: 30 March 2023
Published: 3 April 2023

1. Introduction

Through the contextualization of digital content and information directly into physical reality, Augmented Reality (AR) provides a powerful set of possibilities for training and learning purposes. The added benefits of using AR in education are generally well known, and studies indicate that the application of AR as an additional "multimedia source" in existing curricula can already lead to improved retention, attention, and satisfaction [1]. A meta-analysis shows increased academic achievement with AR compared to traditional learning methods, along with increased concentration, and it also indicates that it enables teachers to convey concepts faster and with greater clarity through the demonstration of connections between concepts and principles [2]. Additionally, secondary literature furthermore points towards a consistently positive impact of AR tools used in educational settings [3], especially through interaction, catching the learner's attention, and increasing motivation [4]. In particular, although significant differences can be observed for all levels of education, the largest effect size of learning benefits can generally be observed for students at the undergraduate level [2]. Therefore, while user attitudes towards AR are influenced by the perceived usefulness and perceived enjoyment of the user, the findings indicate that perceived enjoyment is a more significant factor than perceived usefulness regarding the intention of using AR as a learning source [5].

1.1. Current Challenges in Educational AR

If those benefits are already well known, why is there no widespread adoption of AR in educational settings? While the answer may at least partially be found in the generally

hesitant adoption of information and communication technology in educational contexts because of rigid structures, restrictive curricula, and teachers lacking relevant pedagogical training [6,7], there is also the problem of the availability of a suitable, scalable AR that can be directly applied to aid the teachers' educational goals. While AR hardware in the form of head-mounted devices (HMD), which is increasingly available at lower costs, and commodity handheld/mobile AR hardware, which already comes with solid marker-less tracking techniques, would, in theory, allow for a realistic deployment of AR into teaching curricula today, this currently is not true for the AR software complements. Currently, most AR applications are rather narrow in their scope, often focusing on specific subjects and showing learning benefits with pre-defined prototypes [7], while teachers would primarily need approaches with common concepts, where they would like to be involved in the development process [8]. Therefore, as teachers rarely have relevant AR programming expertise [9], the development of new AR content should be made as easy as possible for them and ideally not involve any scripting or programming [9,10].

1.2. Creating AR Content

This process of AR content creation is generally referred to as "authoring," and it is one of the biggest general challenges in AR today. Even outside of education 10 years ago, Schmalstieg, Langlotz, and Billinghurst [11] already recognized authoring to be one of the five big challenges holding back the widespread adoption of AR. While most of their proposed challenges, such as low-cost platforms, mobility, and suitable back-end infrastructure, are already solved on mobile AR devices utilizing Android and iOS, even today, AR content authoring remains a challenge. There are multiple reasons why this is the case. Firstly, there is an inherently complex need for a tradeoff between the fidelity of possibly created AR applications and the required technical expertise [12]; in other words, either you can create powerful AR scenarios with complex interactions or it can be used with very little technical expertise. Finding the "sweet spot" where non-experts in the technical domain can create complex AR scenarios on their own is still a research question to be answered, so today's AR applications are either built by programmers with significant expertise or authored AR content by non-technical domain experts offering little-to-no interactions [13]. Secondly, AR App development is difficult. A study conducted by Ashtari et al. [14] found that even programmers, ranging from hobbyists to professionals, face consistent challenges with AR development. They report a current lack of concrete design guidelines and examples and incorporation of novel interaction metaphors such as physical aspects to be challenging, and they complain about many unknowns in terms of development, testing, debugging, and user evaluations. Lastly, developing authoring tools as a research topic is a thankless endeavor, or as Nebeling [15] described it for toolkit research in general, a "tricky game". Toolkits are hard to develop, tackle multiple challenges from different disciplines at once, and, in the end, are often hard to publish based on their perceived lack of novelty compared to the time investment necessary to make them realistically applicable for further use in research applications or even curricular usage. This lack of recognition of software artifacts is also a topic of ongoing debate in the educational technology research area [16].

As we therefore believe that only usable, scalable, and self-sufficient comprehensive concepts for AR content can realistically enable AR usage in education, we propose TrainAR, a threefold combination of an interaction concept for procedural task training on handheld AR devices, a didactic framework discussing its implementation as a multimedia source in existing curricula, and an AR authoring tool (See Figure 1) enabling both technical but also nontechnical domain experts to create AR trainings. Therefore, the concept of interaction and the didactic framework of TrainAR have already been published in a separate publication [17], and the contribution of this paper is the TrainAR authoring tool for AR trainings. The TrainAR authoring tool is an open-source extension for Unity that uses visual scripting as its main authoring interaction, while still allowing the usage of the full C# functionality on demand. Therefore, TrainAR would be classified as a high-level

programming tool but also a low-level content design tool based on the digital media authoring taxonomy proposed by Hampshire et al. [12] and allows non-programmers with significant media competency and programmers to utilize it.

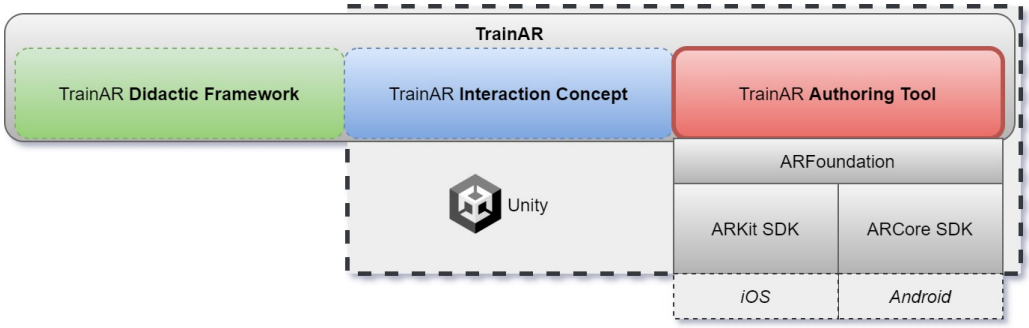

Figure 1. TrainAR: Didactic framework, interaction concept, and authoring tool. This paper discusses the Unity-based authoring tool that is built based on ARFoundation, allowing the deployment to both Android and iOS devices through the ARCore and ARKit SDKs, respectively.

This paper is structured as follows: Section 2 describes related work on AR authoring tools in education and with visual scripting functionality; Section 3 shows the TrainAR components from a technical perspective; Section 4 introduces the TrainAR Authoring Tool; Section 5 introduces current utilization and evaluation efforts; Section 6 describes the utilization and evaluation of currently developed TrainAR Trainings; and Section 7 reports a usability evaluation of the TrainAR authoring tool. Finally, Section 8 discusses the authoring tool and its evaluations, and Section 9 concludes the paper.

2. Related Work

A forthcoming systematic literature review we conducted revealed that comprehensive authoring frameworks comparable to TrainAR, combining interaction concepts, didactic frameworks, and content design tools that could realistically be used to create procedural scenarios by educators, are missing from the literature. Nonetheless, there is some relevant literature on both AR authoring tools in education but also AR authoring tools utilizing visual scripting approaches for the development of more complex AR scenarios beyond 3D objects on AR markers and very basic user actions.

2.1. AR Authoring Tools in Education

An example of a very rudimentary type of educational AR authoring was proposed by Dünser et al. [18] for interactive educational books on physics for HMDs. It utilizes configuration files that could be edited by educators using a standard text editor to connect virtual 3D content with a preset of pictures present in the book. Liarokapis et al. [19] proposed a web-based AR authoring tool with an accompanying client–server architecture that, through the serialization of content into XML, allowed for web-based authoring and viewing of educational engineering content using webcam-based desktop AR and marker-tracking technology to contextualize the content. Laine et al. [20] proposed an AR-based learning games platform, incorporating an AR authoring tool on a desktop PC (personal computer) and an AR viewer on an Android device, that allowed teachers to create interactive, story-driven learning games to increase student engagement in science-related subjects. Blattgerste et al. [21] proposed an authoring tool combining a Microsoft HoloLens and a smartphone as an external controller for the in situ authoring of procedural training tasks, which could be used on HMDs to instruct trainees to perform action sequences. They showed that promptly authored instructions created with this authoring tool could successfully be utilized to help people with cognitive impairments to perform an action sequence they were not previously able to complete [22]. In line with these endeavors, Escobedo et al. [23] utilized a client-server-based AR authoring tool, allowing teachers

to superimpose digital content onto objects to help students with autism to stay focused during learning tasks. Finally, Lytridis et al. [24] proposed a desktop-based authoring tool that allows teachers to contextualize interactive 3D content with images in textbooks similarly to the approach proposed by Dünser et al. [18] but as a web-based approach. Their tool also allows for interactivity by allowing the student to ask questions and receive context-specific learning hints based on the visualized AR content.

2.2. Visual Scripting for AR Authoring Tools

Previous works on visual scripting for AR authoring mostly used either the visual scripting language Scratch, developed in the Massachusetts Institute of Technology (MIT) Media Lab [25], or the JavaScript-based visual scripting abstraction layer Blockly developed by Google. Most of them were used within web-based desktop applications.

Kelly et al. [26] developed a web-based AR authoring tool that utilizes Scratch for the rapid prototyping of real-time tangible user interfaces with AR components. Furthermore, Not et al. [27] utilized Scratch for their cultural heritage museum authoring tool, which also incorporated the possibility to display AR content to visitors. Notably, Mota et al. [28] not only developed a web-based AR authoring tool for the development of educational AR content by teachers through Scratch-based visual scripting, but, through their evaluation, could also successfully show that the block-based visual scripting approach helped teachers to overcome their lack of programming skills and enabled them to develop their own AR learning material combining interactive 3D content with Markers.

Using the JavaScript-based visual scripting abstraction layer Blockly, Apaza et al. [29] developed SimpleAR, a web-based AR authoring tool allowing users to create interactive AR content by selecting objects from the then-available Google Poly 3D database and combining them with AR markers and Blockly-based interactions that could then be used on a dedicated Android viewer application. In line with this, Nguyen et al. [30] proposed BlocklyAR, a Blockly extension allowing AR learners and enthusiasts to author and share marker-based AR scenarios without the need to program.

Most similar to the approach proposed in this paper, Castillo et al. [31] proposed a node-based visual scripting authoring tool that extends the Unity game engine by introducing visual scripting functionality into the normal Unity Editor Layout for the development of educational, marker-based interactive 3D AR content.

3. The TrainAR Framework Components

Forms of interaction with virtual objects and especially procedural interactions in the form of chains of actions are well studied in Virtual Reality (VR) settings and comprehensive toolkits, and frameworks exist to implement them. For example, pre-implemented interaction metaphors and presets delivered with frameworks such as the SteamVR Toolkit, XR Interaction Toolkit SDK, VRTK, OpenVR, or Microsoft MRTK allow developers to focus on the content of their training application rather than worrying about the basic interaction principles with their usability and learnability considerations. For AR, this becomes more challenging, but for HMDs, there are at least gestural interactions, external controllers, and frameworks such as the Microsoft MRTK to provide interaction concepts and basic principles to expand on.

For handheld/mobile AR, the case is even more complicated. While interaction concepts are sparsely explored in the literature and some interaction toolkits do exist, they are currently neither evaluated, nor do they provide the same "out-of-the-box" application utility for developers to directly apply them the same way, compared to VR development. As mobile AR interaction concepts are mostly visual/viewing experiences or ray-casting-based approaches utilizing direct screen touch or UI button approaches, this challenge is only exaggerated in the context of procedural chains of actions necessary for task training. Combining this research gap with the current general challenges faced in Mixed Reality research of finding out how to onboard users to this novel type of application and type of interaction with the uncertainty of how, when, and how much feedback to provide to the

user during trainings, this leads to a substantial amount of time spent by developers on designing interaction concepts from scratch and technical aspects of AR trainings instead of focusing on the content of the training itself. Furthermore, this always requires iterative feedback loops with didactic experts on the interaction and feedback mechanisms. It creates a causality dilemma of not being able to develop a fitting interaction concept and feedback mechanisms without knowing the training task in detail, but also not being able to transfer practical training towards a technically implementable flow of states before having a reference for what such a handheld AR training could look like. This dilemma not only makes AR training development particularly time-consuming for interdisciplinary teams of experts, but makes development of AR scenarios by non-programmers or programmers without AR-specific expertise (e.g., technical-domain experts or designers) impossible.

3.1. TrainAR: From the Interaction Concept and the Didactic Concept towards an AR Authoring Tool

To address these challenges holistically, TrainAR is a threefold combination of (1) an interaction and feedback concept for procedural trainings on mobile AR devices (Android and iOS) that is realistically scalable today, (2) a didactic framework explaining the instructional design theory behind the concepts and how an author should conceptually transfer procedural training tasks into TrainAR trainings, and (3) an authoring tool allowing authors without programming expertise to create TrainAR trainings through visual scripting, based on the interaction concept and didactic considerations (see Figure 1). While the interaction concept and didactic framework were already elaborated, discussed, and evaluated through exemplary implementations in a previous paper [17], this paper focuses on the unity-based visual scripting authoring tool of TrainAR that allows for the creation of trainings in accordance with didactic considerations, utilizing the proposed interaction concept and feedback modalities. Therefore, the following subsection describes the components from the technical perspective as they appear in authored trainings.

3.2. Components of TrainAR from a Technical Perspective

Combining the interaction concept and didactic framework, TrainAR includes concepts and technical solutions for onboarding the user on how to use TrainAR (Section 3.2.1, automatic technical tracking and assembly placement utility, instructing the user on what action to perform next (Section 3.2.3), letting the user perform a procedural non-linear chain of actions (Section 3.2.2), and providing contextualized feedback, insights, and final training assessments aligned with the didactic considerations described in Blattgerste et al. [17] (Section 3.2.3). Those components are included in the authoring tool and automatically included in every training.

3.2.1. Onboarding & Assembly Placement

When an authored TrainAR training is started, trainees are first shown onboarding animations with textual explanations describing how to interact with objects in AR and how to trigger actions on them (see Figure 2a–c). This onboarding utility is included with the framework and automatically deployed by the authoring tool when building the training for a target device. Therefore, no considerations regarding onboarding and concept explanation have to be made by the authors of trainings. When using a training, TrainAR automatically lets the trainee scan (see Figure 2d) a surface area until a sufficiently large free area is recognized by the underlying tracking library and places the TrainAR training assembly onto the surface to start the training (see Figure 2e). This technical tracking and placement utility with its associated onboarding animations is also automatically included in each authored TrainAR training.

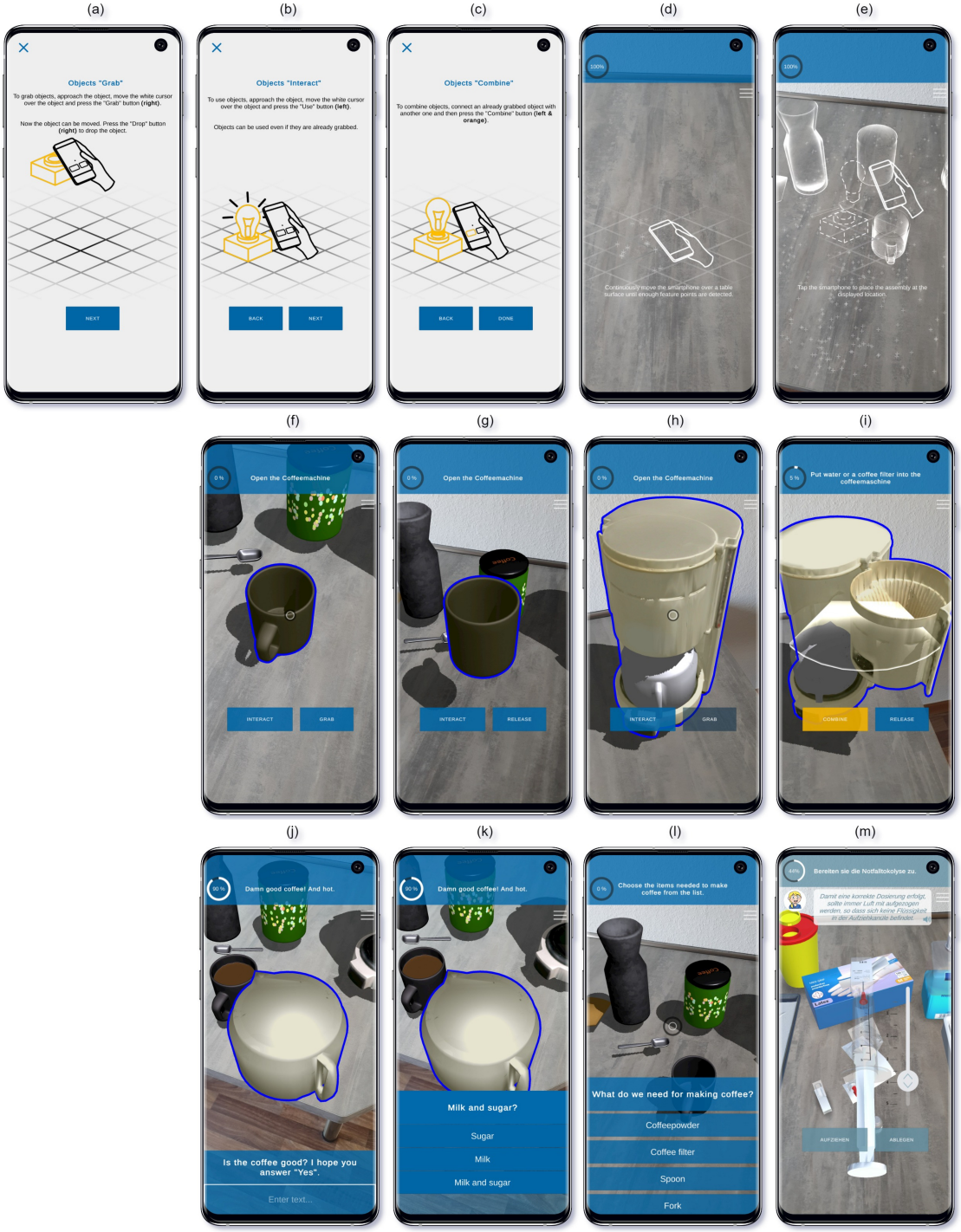

Figure 2. First row: TrainAR automatically includes onboarding screens and technical utility for (**a**) grabbing objects, (**b**) interacting with objects, (**c**) combining objects, (**d**) scanning the training area, and (**e**) placing the training assembly. Second row: The basic AR actions of the TrainAR Interaction Concept that allow trainees to (**f**) select and (**g**) grab TrainAR objects. Selected or grabbed objects can be (**h**) interacted with. Grabbed objects can be (**i**) combined with another TrainAR object by overlapping them. Third row: Custom Actions include UI-based "quiz" actions such as (**j**) text input fields, (**k**) questionnaire elements, and (**l**) list selection elements. Authors can also create their own UI overlays that trigger custom actions, for example, (**m**) a slider to pull up a syringe (see Section 6.1).

3.2.2. TrainAR Objects and Procedural Chains of Actions

After the placement of the training assembly, the trainee can complete a procedural chain of actions defined by the author of the training. Those trainings consist of the basic actions of selecting (see Figure 2f), grabbing (see Figure 2g), interacting with (see Figure 2h), and combining (see Figure 2i) virtual AR objects called "TrainAR Objects".

These TrainAR Objects are virtual AR 3D models that were converted by the TrainAR authoring tool and enriched by scripts, providing them with consistent basic interactive functionality. These automatically inherited responses are as follows. When selecting a TrainAR Object, subtle shading and outlining of the selected object is applied (see Figure 2f). When grabbing a TrainAR Object, it leaps and attaches itself into a static position in front of the handheld device, where it is always rotated into an upward position from the assembly ground and keeps a defined distance from the device to make the object stay visible on the screen throughout the interaction. Users can then displace it, interact with it, or combine it with another stationary object. When interacting with and combining objects, outlines visualize the current state-change of the object and whether the action was accepted (valid) or not, while object-specific interactions that are triggered are defined by the author of the training.

Alongside those basic actions, trainings can have Custom Actions (see Figure 2m) that serve as customizable action triggers defined by the author. This allows authors to implement independent concepts outside the interaction scope provided by TrainAR. Furthermore, trainings can utilize predefined UI components such as input fields (see Figure 2j), questionnaires (see Figure 2k), or list selections (see Figure 2l) to realize in-procedure quizzes or material selection, or to check for decision procedures that could not otherwise be sufficiently covered by just the basic actions of TrainAR.

3.2.3. Instructions, Insights, and Feedback

Besides the actions, which serve as input from the trainee, several types of output modalities are delivered with TrainAR. They are used to elicit instructions, feedback, and insights or to indicate technical problems. Firstly, the technical feedback screens are always included when deploying a training. They automatically trigger when technical problems are detected to provide feedback to trainees, e.g., if there is insufficient light, not enough feature points for tracking, or the smartphone moves too fast (see Figure 3a). Authors do not have to develop any technical instructions or problem feedback themselves.

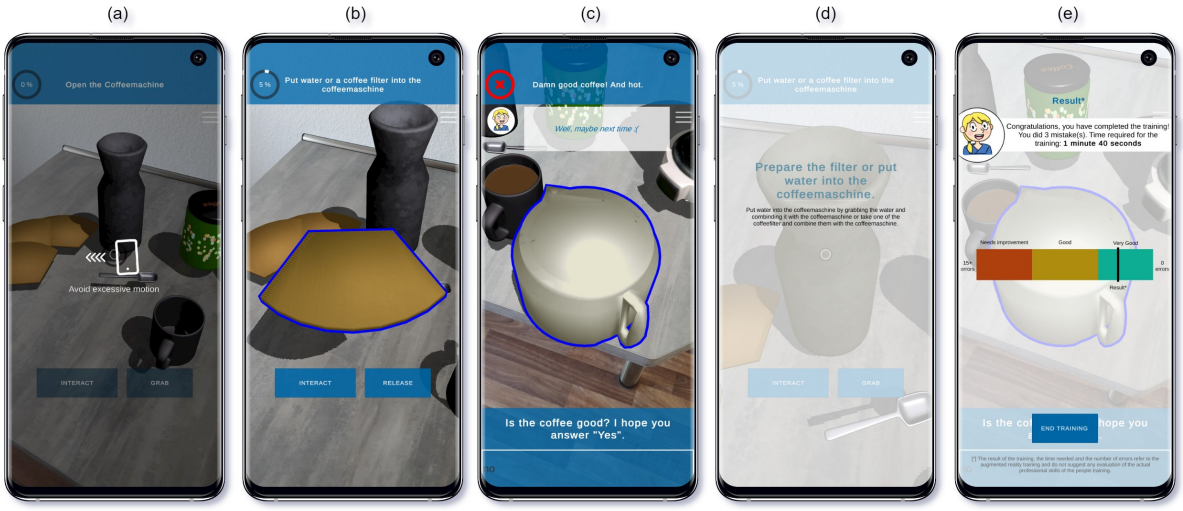

Figure 3. The output modalities of the TrainAR Interaction Concept consist of (**a**) feedback to aid technical problems, (**b**) instructions and progress indicators, (**c**) expert tips and insights, (**d**) error feedback overlays, and (**e**) a training summary at the end of each training.

To instruct the trainee on what action or bundle of actions should be performed next, textual instructions are displayed on a UI panel on top of the device screen, including a progress bar showing the current completion percentage of the training to the trainee (see Figure 3b). After triggering one of the actions, the trainee is always provided with feedback in the form of a blinking outline and a sound effect. For some errors, it might be necessary to communicate a message. In this case, error overlays take the trainee out of the training context into the UI and can show textual feedback to the trainee who has to be dismissed manually (see Figure 3d). Occasionally, there might be information that is neither instruction nor feedback on an action of the user but still important as part of the training, e.g., expert insights that provide additional tips from practice. In this case, insights can be used that display textual tips as a speech bubble UI element at the top of the screen, optionally also including auditory tips (see Figure 3c). After the training is concluded, a training summary is displayed to the trainee showing the training time, number of errors, and the errors contextualized on a performance scale (see Figure 3e).

4. The TrainAR Authoring Tool

The TrainAR Authoring Tool is a Unity-based authoring environment that is built upon the Unity Editor interface and utilizes the ARFoundation, ARKit, ARCore, and Visual Scripting packages. Its layout inside Unity is displayed in Figure 4. It allows authors to create TrainAR procedural trainings out of the components described from the technical perspective in Section 3.2, utilizing the interaction concepts and didactic perspective proposed in [17]. The authoring tool thereby delivers the interaction concept and all action implementations, feedback mechanisms, and technical solutions for onboarding, tracking aid, and training assembly placement. Furthermore, it provides tools to convert 3D objects into TrainAR Objects that automatically inherit all TrainAR behaviors necessary to work within the flow of states of the training. The author of a TrainAR training only has to import 3D models, convert them, and then reference them in a procedural visual scripting flow to specify their state changes during the training based on user actions. Authors can then optionally implement additional guiding instructions, feedback modalities, or quizzes. These two central concepts and the remaining tasks for the authors are referred to as the "TrainAR Objects" in the "Training Assembly" and the "TrainAR Stateflow" in the "TrainAR Statemachine" (see Figure 4). To enable authors to accomplish those tasks, the layout of the authoring tool is split into several regions: the "Unity Project folder" that shows all imported Assets in the project, a simplified "Unity Inspector" with a list of Objects currently displayed in the scene, the "TrainAR Assembly Scene", allowing authors to view the TrainAR Objects of their training contextualized on a reference setup, and the "TrainAR Visual Statemachine", which allows authors to determine the state flow during the training, based on the users' actions. The "Device Preview" allows authors to preview the training assembly from the perspective of the users' smartphone.

At the time of publication, the TrainAR Stateflow encompasses 10 types of visual-scripting nodes that can be used by referencing the corresponding TrainAR Object by name. All included nodes are visualized in Figure 5 in relation to the interaction concept [17]. They are described in more detail in the TrainAR online documentation. The *TrainAR: Onboarding completed, and training assembly placed* node indicates the start of the TrainAR training and automatically starts the flow of states after the Training Assembly was placed in AR by the trainee. The *TrainAR: Object Helper* node is a collection of tools that help to change the state of TrainAR Objects when reached during the flow of states, e.g., changing their visibility, possible actions this object responds to, or replacing them with other objects during the flow. Additionally, four of the nodes are action nodes. If the TrainAR Statemachine reaches one of these nodes during a TrainAR training, it waits for an action by the trainee. These actions can be grabbing, interaction with or combining TrainAR Objects. This can either be exactly one specific action to continue (*TrainAR: Action*), n multiple actions in no particular order (*TrainAR: Action (Multi)*), n actions that lead to $m <= n$ different flows of actions as a consequence(*TrainAR: Action (Fork)*), or the requirement for the user to complete a quiz

such as a questionnaire, list-selection task, or text input (*TrainAR: Action (UI)*). The four remaining nodes are output and feedback nodes. The *TrainAR: Instructions* node allows the author to provide adaptive textual instruction to the user of the training. If the user should be provided with specific feedback during the training when performing an incorrect action, the *TrainAR: Feedback* node can be used. Sometimes, information has to be conveyed that is neither direct instruction on what action to perform next, nor feedback based on a performed action. In this case, *TrainAR: Insights* can be used to, e.g., provide additional tips or insights from practice to the trainee.

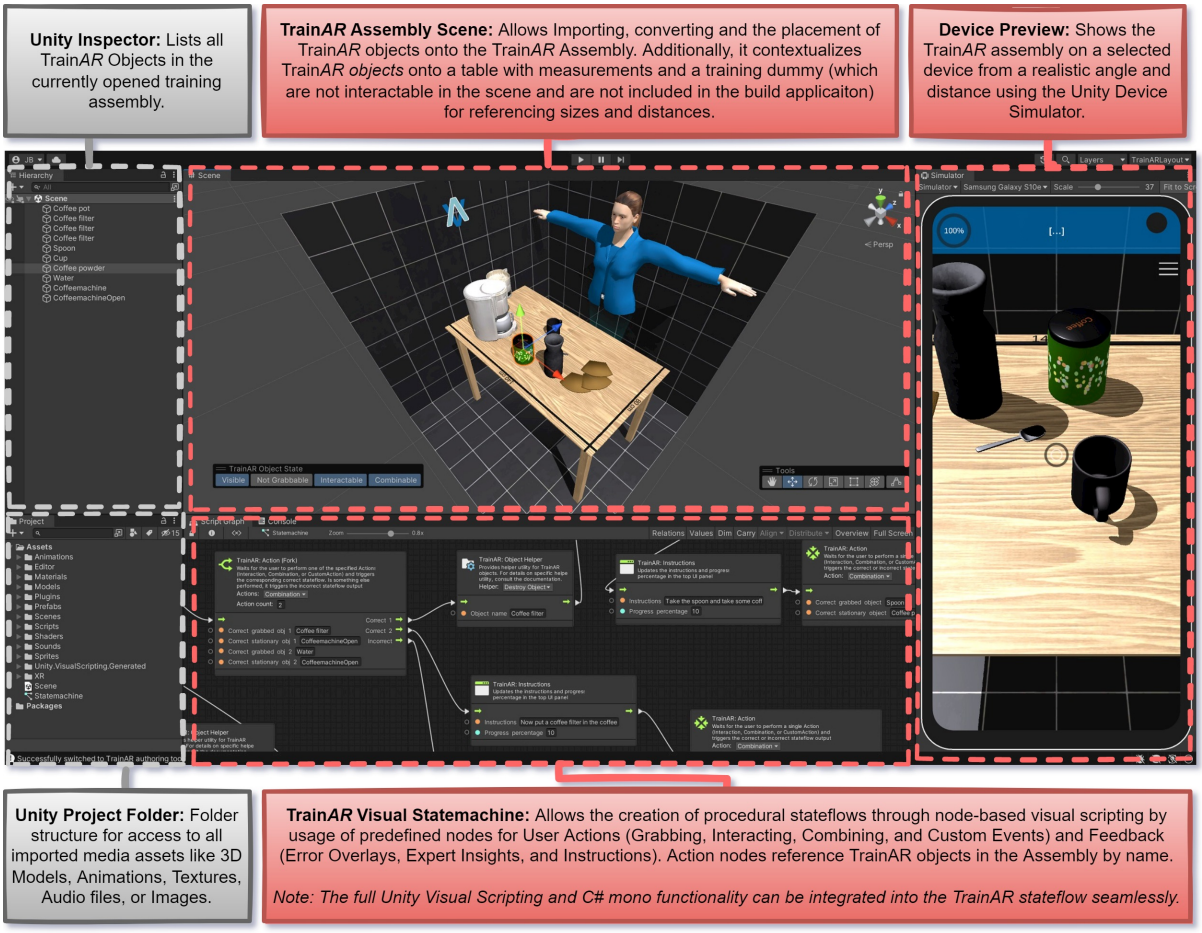

Figure 4. The TrainAR Authoring Tool layout combines the Unity Inspector and projects folder with the TrainAR Training Assembly, TrainAR Visual Statemachine, and a Scene preview, allowing authors to create procedural TrainAR trainings.

4.1. Design Considerations for the TrainAR Authoring Tool

According to Hampshire et al. [12], AR authoring tools can generally be classified into low-level programming tools, high-level programming tools, low-level content design tools, and high-level design tools. With the increasing abstraction of concepts, authoring tools can also use higher-level interface abstractions, which makes them easier to use. However, as a consequence, this also increasingly limits the AR scenarios that can be created with the tool. Although it would seem plausible to try to target teachers themselves and, therefore, design a high-level content design tool, we deliberately designed and developed a low-level content design tool with TrainAR. While higher-level standalone approaches were considered during the conceptualization, this decision was made for two reasons.

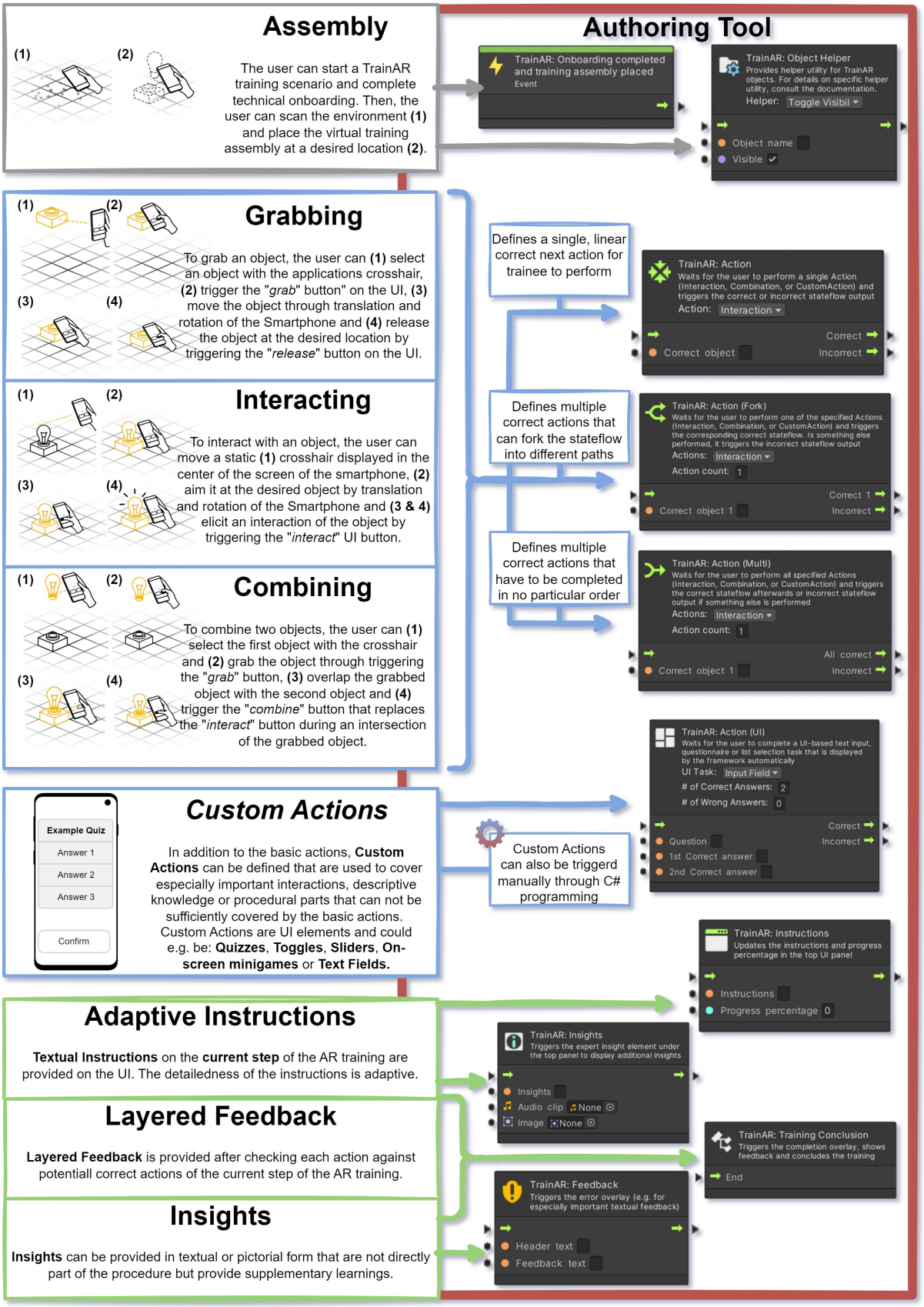

Figure 5. The concepts introduced in the TrainAR interaction concept and didactic framework (**left**) and their corresponding Statemachine nodes in the TrainAR Authoring Tool, which can be used to author the TrainAR training.

Firstly, even if the authoring tool itself was designed as a high-level content design tool, the generation, or acquisition of 3D assets to use in the authoring tool would still likely require significant media competency (described in more detail in Section 4.4), possibly nullifying the gained advantages from the higher interface abstractions, and could even be too time-consuming to be realistically performed by educators/teachers themselves.

Secondly, this approach allows us to implement the TrainAR authoring tool as a Unity extension, which provides several advantages but inherently comes with increased complexity of the user interface of the tool. As such, TrainAR is an abstraction layer, which allows creating and deploying AR Trainings without any programming expertise (a low-level content design tool), but as an extension, it is also fully integrated into the C# environment and Unity's own Visual Scripting approach. With this approach, programmers can also use TrainAR as a starting point or high-level programming tool and expand it where necessary (described in more detail in Section 4.5).

4.2. Open-Source Availability and Documentation

The complete source code of the TrainAR authoring tool is available as a Git Repository under https://github.com/jblattgerste/TrainAR/ (Accessed: 31 March 2023) under the MIT License. Besides the full source code for the authoring tool as a Unity Editor extension, this includes the complete source code for the TrainAR interaction concept, a full documentation of the code, API references (see https://jblattgerste.github.io/TrainAR, accessed: 31 March 2023), and a "Getting Started Guide" (see https://jblattgerste.github.io/TrainAR/manual/GettingStarted.html, accessed: 31 March 2023) that helps authors of TrainAR trainings to quick-start their AR training development. Additionally, it helps programmers to expand TrainAR towards context-specific needs in a dedicated section to expanding TrainAR.

4.3. Envisioned Workflow for Authoring TrainAR Trainings

The designed workflow of using the TrainAR authoring tool is described in detail in the "Getting Started Guide" for one example scenario. Abstractly, it is envisioned as follows.

First, the user downloads the Unity Editor and installs it on a Windows, Linux, or macOS computer. Afterward, the user can download the TrainAR project from GitHub either as a .zip folder or by cloning it via git. Opening the project in the specified Unity version allows the author to then switch Unity to the TrainAR authoring tool mode through a context menu, providing the author with the authoring tool setup shown in Figure 4.

The user can then start with the authoring of a TrainAR scenario by importing 3D models into the Unity Project Folder, placing them into the TrainAR Assembly Scene, and converting them into TrainAR Objects through a simple click on a button that starts TrainAR's model conversion process visualized in Figure 6. Afterward, through overlays in the TrainAR Assembly scene, the author can then translate, rotate, and scale models and define the TrainAR objects' initial set of interaction abilities (Visible, Grabbable, Interactable, Combinable). The author can thereby compare sizes and distances based on the reference preview scene provided with the authoring tool. After the conversion of all models and the arrangement of the training assembly scene, the author can create the flow of states in the TrainAR Visual Statemachine through visual scripting. This is carried out by adding the TrainAR logic nodes specified in Figure 5 and referencing TrainAR Objects in those nodes by name.

After completing the authoring process of both the TrainAR Stateflow and assembly, the author can connect an Android or iOS device to the computer and press the Play button at the top of the editor to install the TrainAR training app to a smartphone. This deploys the training to the device and, besides the authors' objects and stateflow, automatically includes the TrainAR interaction concept, onboarding animations, technical tracking, and assembly placement utility.

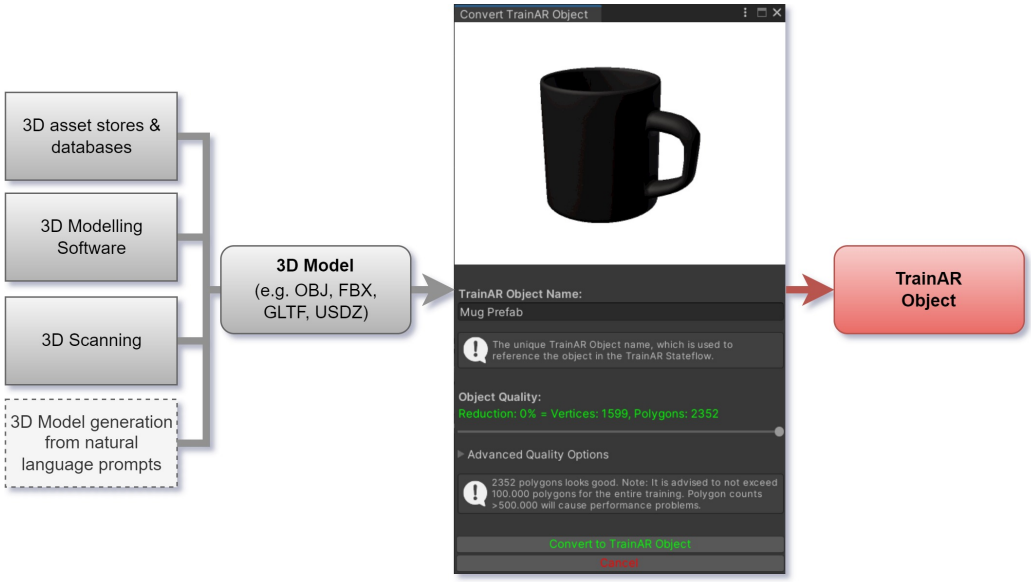

Figure 6. The modal window for the conversion of 3D objects from a variety of sources and in different formats into a consistent TrainAR Object, which is simplified, compatible with TrainAR, automatically inherits all intended behaviors, and can be referenced in the TrainAR Visual Statemachine.

4.4. Content Generation through 3D Scanning and Natural Language Prompts

With this authoring workflow, the most challenging technical aspect remaining for authors is likely the generation of the 3D content for the AR trainings. While there is an increasing availability of educational 3D content on the web [32], available models might not always be fitting or there might be no models available for specific training contexts. Therefore, besides the creation of models through 3D modeling software such as Blender, or the processing of CAD models, a central consideration for TrainAR is the generation of 3D content through 3D scanning. Through pre-checks and mesh conversions during the conversion from normal Unity GameObjects with attached 3D meshes in various formats to TrainAR Objects, models are created automatically that are compliant with the TrainAR framework, independent of their source and initial structure (see Figure 6). As this includes mesh reparation, simplification, and merging, this is not only helpful for 3D scanned objects but also paves the way for the inclusion of meshes from other sources that will emerge in the near future, e.g., 3D models generated through natural language prompt-based approaches, as is currently being researched by Google Research [33].

4.5. Beyond TrainARs Statemachine: Expanding on the TrainAR Framework

In anticipation that the TrainAR Statemachine, while being the factor that enables non-programmer domain experts to utilize it, would also be the most limiting factor for more experienced users and programmers trying to implement more context-specific requirements, we deliberately chose to develop TrainAR's authoring tool in the form of a Unity extension and expanded upon the Unity Visual Scripting Package [34] for the visual TrainAR Statemachine.

This approach allows for an expansion of the TrainAR authoring tool in several directions. Foremost, the custom Action node allows for triggering state changes with a parameter from a MonoBehaviour manually, allowing for user actions besides grabbing, interacting, and combining out of the box. Additionally, nodes provided by Unity's Visual Scripting package are completely compatible with all TrainAR nodes, making it possible to integrate them into stateflows for more complex behaviors in terms of the flow of actions, therefore providing stateflow-level expansion possibilities for TrainAR. For the expansion of TrainAR on the object level, when switching into the Unity Editor layout, all MonoBehaviours of converted TrainAR objects are exposed, and object-level Events, e.g., for this specific object being selected, interacted with, grabbed, or combined, are exposed

as UnityEvents and can be used to implement more complex object-level behaviors such as animation triggers or object-specific MonoBehaviour C# scripts that trigger event-specific custom behaviors. Finally, if authors want to use the interaction concept and technical onboarding utility of the framework in non-procedural training contexts, e.g., for conceptual training games, rule-based stateflows, or simply want to program stateflows themselves, the visual statemachine can be switched off entirely by simply commenting out a single line of code in the Statemachine connector (see https://github.com/jblattgerste/TrainAR/blob/main/Assets/Scripts/Static/StatemachineConnector.cs, accessed: 31 March 2023) and handling the requests of the state change function manually through C# scripting.

5. Utilization and Evaluation

Evaluating TrainAR's authoring tool inherently means evaluating TrainAR holistically as a framework for the creation and utilization of digital, procedural AR trainings. Consequently, this requires the evaluation of several of its components individually, making the evaluation challenging and extensive. Additionally, simple lab studies with preliminary prototypes would likely not suffice to evaluate TrainAR's most important aspects, or might even be misleading based on our perspective and usage vision. While extensive evaluations are ongoing, the following Sections 6 and 7, provide preliminary insights into our current results. We believe that four questions have to be answered from the perspective of somebody trying to utilize TrainAR to entice them to apply it to their context:

1. Do TrainAR trainings elicit learning benefits (e.g., increased retention, conceptual understanding, or motivation, or providing self-paced learning opportunities)?
2. Are TrainAR trainings usable by and enjoyable for the trainee?
3. Is the TrainAR Authoring Tool usable by non-programmers to create such TrainAR trainings? More specifically,
   (a) What is the required level of media competency, and who can realistically utilize the TrainAR authoring tool?
   (b) How fast can the usage of the tool be learned, and what training or tutorial material is necessary?

These questions are in line with the accepted User Experience design principle *Utility* + *Usability* = *Usefulness*, which conveys that a product has to provide utility and be usable by the target group to be a useful product. We believe, in this specific case of the AR authoring tool, that this principle has two levels (see Figure 7). First, the Utility and Usability of the TrainAR training has to be shown to prove them to provide a useful training. If this is true, the utility of the AR authoring tool would consequently be a possibility for creating a useful AR training. Then, it has to be shown who can use the AR authoring tool to create these trainings, as usability is as dependent on the target user as it is on the implementation.

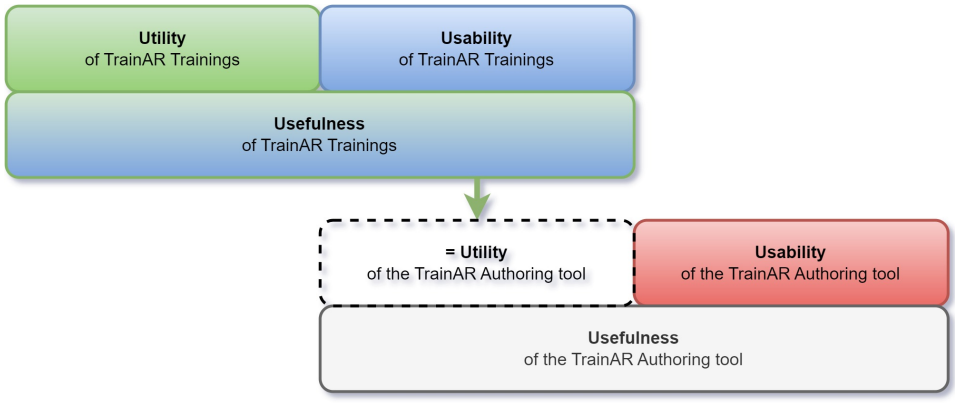

Figure 7. To show the usefulness of the TrainAR authoring tool, its utility and usability have to be evaluated. The utility of the authoring tool itself is the creation of trainings that themselves have to be useful, meaning they also have to prove their utility and usability.

6. Utility and Usability of TrainAR Trainings

To answer research questions 1 and 2, several TrainAR trainings are currently in development or were developed and evaluated using the TrainAR framework in different contexts. While evaluations have not concluded for all the trainings, five exemplary TrainAR Trainings are shown in Figure 8, and their utility and usability evaluations are described below. For usability assessment, the System Usability Scale (SUS) was used to make results comparable across the trainings (see Figure 9). As the desired utility of each TrainAR training and its evaluation is highly dependent on the context, we only discuss the utility on an abstract level and refer to the authors' publications for more detailed insights and discussions.

Additionally, although TrainAR was originally envisioned as a holistic solution combining an interaction concept, didactic framework, and an authoring tool (see Figure 1), as already discussed in [17], each of the three components can also be used separately. It has to be noted that the scenarios shown in this section do not necessarily use each of the components. While all of them use the TrainAR interaction concept, the training of preparing a tocolytic injection was the starting point for the TrainAR framework abstraction and therefore was developed from scratch, not utilizing the TrainAR authoring tool. The denomination of the female pelvis, a game exploring the sourness of fruits, and the game for exploring ripeness all use the interaction concept and authoring tool but not the didactic framework, as they are envisioned more as rule-based learning games than strictly procedural trainings. Only the titration experiment utilizes all three components, though it also has to be noted that the authoring tool utilized in all trainings was in early preliminary stages, e.g., not including visual scripting and still requiring programming (see Table 1). The main focus in this stage was the evaluation of the created trainings, not the authoring tool, which was evaluated separately (see Section 7).

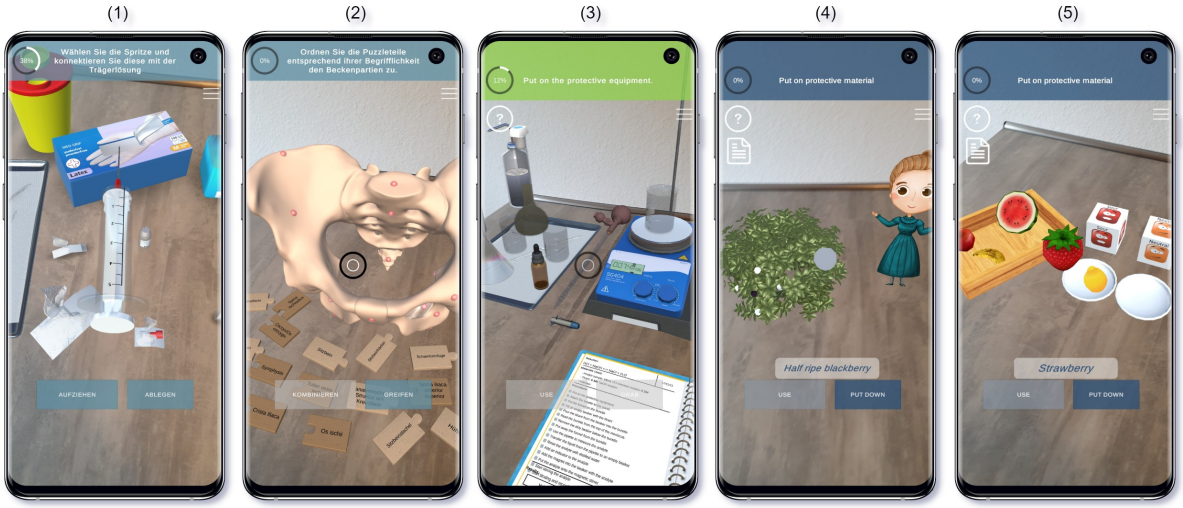

Figure 8. Five exemplary TrainAR trainings. (**1**) The preparation of a tocolytic injection in the context of academic midwifery, (**2**) the denomination and contextualization of German and Latin terminology of the female pelvis, (**3**) a titration experiment in the context of chemical engineering, (**4**) the exploration of chemical reactions in early school education, and (**5**) the exploration of ripeness as a chemistry learning game for children.

Table 1. The five exemplary TrainAR trainings shown in Figure 8 and which parts of TrainAR (Interaction concept, didactic framework, or authoring tool) they utilize for their use case.

TrainAR Scenario	Interaction Concept	Didactic Framework	Authoring Tool
(1) Preparation of a Tocolytic Injection	✓	✓	
(2) Denominating the Female Pelvis	✓		✓
(3) Conduction of a Titration Experiment	✓	✓	✓
(4) Exploring Chemical Reactions	✓		✓
(5) Understanding Fruit Ripeness	✓		✓

6.1. Preparation of a Tocolytic Injection

In the context of the Heb@AR project, a procedural TrainAR training was developed for the preparation of a tocolytic injection in the context of academic midwifery education (see Figure 8(1)). Here, the user elicits a sequence of actions to prepare a tocolytic syringe that is labeled and stored in a fridge, which is a common task in the daily midwifery routine [35]. For this purpose, the user has to interact with objects and grab, place, and combine objects while being instructed and tutored by a virtual professional midwife [17].

The desired utility of the training is an opportunity for self-directed, location-independent learning and an increase in self-efficacy for midwifery students [35]. While the results for the evaluation of the utility are forthcoming, preliminary analyses and qualitative feedback look promising. To measure the perceived usability of the training, the SUS questionnaire was used. A SUS study score of 83.11 (SD = 12.9) was reported (see Figure 9), which would indicate "Excellent" usability according to Bangor et al. [36] and surpasses the non-empirical, but commonly used, industry benchmark of SUS study scores of 80 [37]. With a sample size of $n = 33$ participants, the results are 100% conclusive, according to Tullis et al. [38].

6.2. German–Latin Denomination of the Female Pelvis

Likewise, in the midwifery education context of project Heb@AR [35], a learning game for the denominating of the female pelvis [39] was developed (see Figure 8(2)). Here, the idea is to use TrainAR's gamification aspects to make the traditionally dry subject of learning all German and Latin names and their contextualization for the bones and regions of the female pelvis more enjoyable to students. The user has to grab and combine pieces of a puzzle with the Latin and German names with each other, and then contextualize them to corresponding bones and regions of the female pelvis.

In terms of utility, it was found to increase the students' intrinsic motivation to engage with the historically dry subject significantly, which was measured through a within-subject comparison using pre- and post-study questionnaires [39]. For usability, a SUS study score of 84.79 (SD = 13.51) was reported (see Figure 9). This would not only be interpreted as "Best Imaginable" Usability according to Bangor et al. [36] and surpass the non-empirical industry benchmark of 80 [37], but is also the highest recorded SUS study score of a TrainAR training recorded to date. The sample size of $n = 36$ participants should yield 100% conclusive results, according to Tullis et al. [38].

6.3. MARLabs Titration Experiment

In the context of academic chemical engineering education, Dominguez Alfaro et al. [40] from KU Lueven developed a TrainAR procedural training where students, preparing for their actual physical lab titration experiments as part of the curriculum, can train the necessary procedures of titration experiments beforehand. They can use their smartphone to combine chemicals, follow safety procedures, and document their experiment accordingly (see Figure 8(3)).

For the usability, a SUS study score of 72.8 (SD = 14.0) [40] was reported (see Figure 9), which would indicate above average or "Good" usability on the Adjective contextualization scale proposed by Bangor et al. [36] and is an acceptable usability score [41]. According to

Tullis et al. [38], this result is only between 75–80% conclusive, based on the small sample size of $n = 9$ participants. The desired utility of the training was an increased understanding of the users' knowledge of acid–base titration concepts. Likely because of the small sample size, the initial study failed to show statistically significant learning effects, but results from larger studies are forthcoming [40]. Nonetheless, Dominguez Alfaro et al. [40] could observe that the app was "well-received by the users", and they were able to independently download and utilize it in a remote experiment setting without an experimenter present.

6.4. Exploration of Fruit Ripeness and Sourness

Finally, for the K-12 chemistry education context, learning games for the exploration of fruit ripeness (see Figure 8(4)) and exploration of the sourness of fruits (see Figure 8(5)) were developed for iOS tablets by Arztmann et al. [42] at Utrecht University, using TrainAR. Dutch children aged between 11 and 15 used the application as part of their curriculum to have playful first points of contact with chemical principles such as ripeness and sourness by, for example, feeding beets with different levels of ripeness to a virtual avatar or analyzing fruits based on their sourness, using a pH strip, and then sorting them.

The intended utility of the training was the possibility for students to independently engage with these new concepts playfully and at their own pace. Therefore, the idea was "triggering students' interest in chemistry by providing a playful environment with relatable content" [42]. While this is challenging to quantify, it was observable that the students were able to independently utilize the game and were enjoying the experience. A non-validated Dutch translation of the simplified SUS questionnaire by Putnam et al. was used [43] to measure the perceived usability for this usage group. The resulting Dutch simplified SUS questionnaire had low internal consistency, with a Cronbach's alpha of 0.446. The calculated SUS study score of 54.7 (SD = 15.19) (see Figure 9) would be interpreted as "OK" [36], but below average, perceived usability and indicates only "marginally acceptable" usability according to Bangor et al. [41]. Besides the internal reliability issues, a SUS study score with $n = 239$ participants should be conclusive based on the sample size, according to Tullis et al. [38]. With low internal reliability, children as the target group instead of adults, and the usage of iOS tablets instead of smartphones as the delivery method, it is hard to determine where this low perceived usability, compared to the other TrainAR trainings, originates. It might be possible that, children, who are not the originally envisioned target group [17], require additional considerations [42]. Additionally, interaction effects are possible. These perceived usability results should therefore be interpreted with caution.

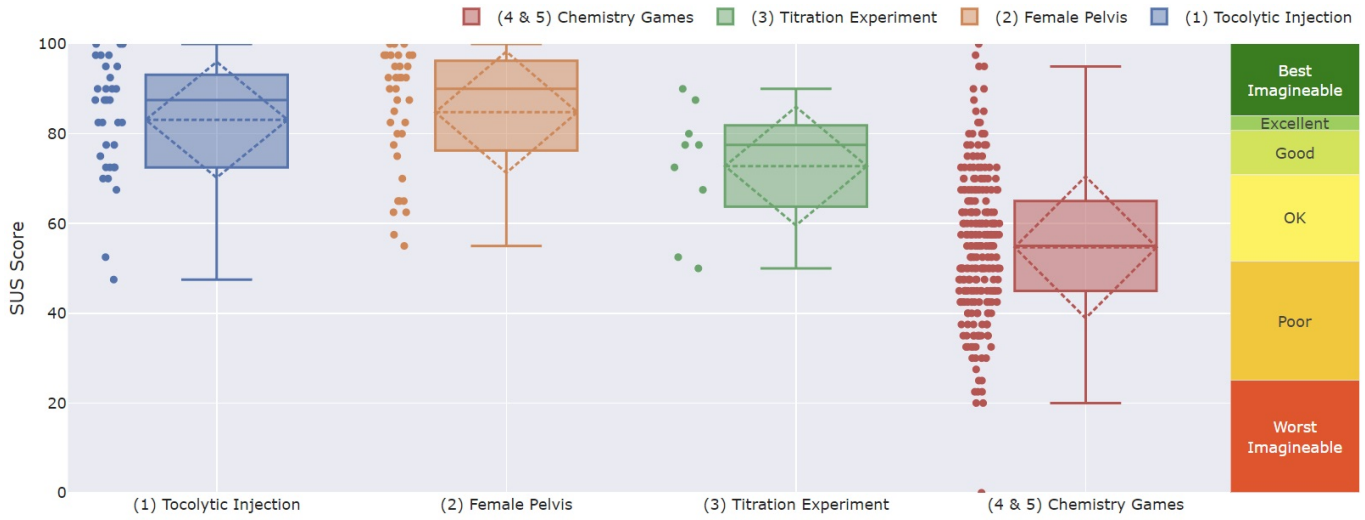

Figure 9. The perceived usability of the exemplary TrainAR scenarios in Figure 8 in the form of SUS study scores, taken from primary sources evaluating TrainAR trainings [17,39,40,42] and plotted with the SUS Analysis Toolkit [44].

7. Usability of the TrainAR Authoring Tool

To answer research question 3, we carried out multiple steps. First, we shared the framework with researchers from Utrecht University and KU Leuven in 2020 for them to deploy it in their contexts; then, we iteratively used the authoring tool in two practical lectures to observe its usage in a non-representative setting. After this indicated sufficient maturity of the authoring tool, we conducted a systematic study to determine the tool's usability and to assess the required media competency.

7.1. Pre-Study and Non-Representative Observations

Initially, we shared early versions of the TrainAR framework with other Universities in 2020 to deploy them to their contexts. During this process, the trainings described in Section 6 were created. The TrainAR versions used by those collaborating researchers were early builds, e.g., not including the Visual Statemachine and providing a less convenient object-conversion utility. The researchers, while not computer scientists, had experience with programming. While this provided valuable first insights into the feasibility of TrainAR's set of utility and the effectiveness, usability, and enjoyability of the authored TrainAR trainings, these insights were not representative of the usability of the authoring tool itself.

Afterward, early versions of the authoring tool, then already including the full TrainAR Visual Statemachine functionality, were used during the practical part of an apprenticeship course to obtain preliminary insights into the usage of TrainAR by the main target group of the framework: domain experts with high levels of media competency but without programming knowledge. In this course, eight apprentices created four TrainAR scenarios of their choice through the course of four practical sessions, each lasting around 2 h. Throughout this course, the apprentices chose to create scenarios for the installation of a desktop computer set, the finishing work after 3D printing mechanical components, cutting and filing a workpiece, and the preparation of a steak with bacon and eggs. Besides some smaller hurdles and anticipated bugs, which could be either resolved by consulting the teaching assistants present or through smaller technical adjustments to the source code of the framework, the apprentices were able to create procedural action chains using TrainAR's authoring tool. They were even able to incorporate 3D scanning for model generation. The observations and feedback provided showed that the authoring tool was sufficiently usable for them. This indicates that it should also be usable for the envisioned target group and that it is possible to independently create TrainAR trainings for them. The most challenging aspect was the 3D model generation or gathering and the didactically conceptual, but not technical, chaining of instructions, actions, and feedback mechanisms. Their feedback furthermore highlighted that good documentation and especially in-depth onboarding and "Getting Started" utility would be helpful. After improving the documentation, the conversion utility, fixing bugs in the source code, and publishing TrainAR on GitHub, this procedure was repeated with another course of 12 students, and then six scenarios were created over the course of four practical sessions that were 2 h each. Here, students were again able to successfully create trainings with the authoring tool.

7.2. Systematic Usability Study Design

After the second pre-study iteration was successful and indicated that most major problems had been addressed, we conducted a systematic usability study, with a focus on the pragmatic qualities of the authoring tool and the required media competency to use the TrainAR Authoring Tool. While ideally the usability evaluation would be conducted with actual users of the authoring tool and correlations between the pre-existing media competency recorded through standardized tests would be investigated, this is challenging in practice for multiple reasons. Firstly, actual users are challenging to recruit for the study, because of resource and time constraints. Then, those users would have to be recruited systematically and in high numbers, so there are actual differences in media competency. Finally, systematic assessments of media competency often rely on self-

reported measures and mostly focus on computer literacy, which would likely not be sensitive to differences in the media competencies we are interested in. Therefore, the study was designed as a between-subject comparison with students as participants from three groups: Computer Science (CS) students, Media Technology (MT) students, and non-technical students from our university. Before the experiment, we asked participants to self-assess their competency in 3D modeling, programming, and VR/AR/Game development. During the study, participants had to author three trainings based on provided 3D models and stateflow descriptions in line with task process analyses, totaling 47 sub-tasks to complete the study. The 47 sub-tasks consisted of 10 types/categories of tasks, e.g., placing or converting an object, placing an action node in the Visual Statemachine, or placing an instruction node. The three authoring tasks hereby increased in complexity, with the first one (mounting a lightbulb in a socket in order to subsequently switch it on) being a simple, linear flow of actions, the second task (a re-enactment of the East Frisian tea ceremony) introducing quizzes and UI elements, and the third task introducing non-linear flows of states (attachment of a needle to a syringe and subsequently filling it with medication). During the experiment, we recorded the Task-Completion Time (TCT) and Task-Completion Rate (TCR) for each of the 47 sub-tasks and cognitive load (NASA rTLX) [45] after each of the three tasks. After the experiment, we measured the perceived usability using the System Usability Scale (SUS) [46], which is one of the most widely used usability questionnaires and provides direct benchmarking and contextualization utility [44]. Finally, we asked the subjects for qualitative feedback to self-assess their ability to independently create AR trainings using TrainAR.

7.3. Setup and Procedure

A desktop computer (AMD Ryzen 7 5800X, 64 GB Ram, Nvidia GeForce 3080 Ti) with two 30-inch monitors and a stand microphone was used for the experiment. The experimenter was sitting beside the participants during the study and took notes about TCT and TCR, and audio was recorded during the experiment.

After greeting the participants and explaining the study, they were asked to fill out a pre-study questionnaire. Here, they filled out a declaration of consent, a demographic questionnaire, and a questionnaire on their relevant previous knowledge. Afterward, they were given a brief introduction to TrainAR in the form of a 4-min explanation video. This video explained the basic features and functionalities of TrainAR and its general use. Furthermore, the participants were given a short verbal introduction to the documentation of TrainAR and were encouraged to use it during the study. Then, participants used the TrainAR Authoring Tool to implement the three pre-defined TrainAR trainings. The tasks to create the trainings were divided into 47 sub-tasks, which were presented to the participants sequentially in a Google Forms document. For each of the sub-tasks, the desired end result was shown either as a short video clip or an image and their intended connection in the flow of states as a task-process-analysis-inspired visualization. The participants then had to author each sub-task on their own. In case they needed help, they were allowed to ask the experimenter for hints or help. These hints and the help were given systematically on four levels. Each of the sub-tasks had a documentation hint, meaning a pointer to the relevant passage in the TrainAR documentation. This hint was given as a first measure, should the participant run into problems. If this hint did not help, the participant was provided with a solution hint, meaning a pre-defined hint for the given sub-task that was read out to the participant. If participants were unable to solve the sub-task with this hint, they were explicitly helped by the experimenter. This ensured that each participant was exposed to all 47 sub-tasks during the experiment, which built upon each other. After a participant completed all sub-tasks of this authoring task, they were asked to test the scenario on a provided smartphone and then filled out a post-task questionnaire, containing a perceived cognitive workload questionnaire (NASA rTLX) [45] regarding the just-completed scenario. Before starting with the authoring of the next training, the participants were offered a short break and snacks. After all three authoring tasks had been completed in this fashion, the

participants were finally asked to fill out the post-study questionnaires, containing the SUS, as well as a qualitative feedback questionnaire asking them what they liked, where they had problems, and to assess if they would be able to create an AR training independently.

7.4. Participants

Overall, 30 participants took part in the experiment. Twenty-one of the participants were male, and nine were female. The average age of the participants was 25.13 (SD = 3.24). Participants received monetary compensation for their participation in this study.

The participants were recruited from groups: 10 participants were Computer Science (CS) students, 10 were Media Technology (MT) students, and the remaining 10 students were from various non-technical study programs (i.e., business administration and social work studies) from our university. To validate if their self-reported competency matched our expectation of the groups, participants were asked to rate their experience in 3D modeling, programming, and VR/AR/game development on a seven-point Likert scale ranging from 1 ("no experience at all") to 7 ("very experienced"). CS students reported the highest experience in both programming (M = 5.40, SD = 1.11) and VR/AR/game development (M = 3.60, SD = 2.01), indicating they were experienced in programming and somewhat experienced in VR/AR/game development. MT students reported that they were somewhat experienced in programming (M = 3.00, SD = 1.18) and not at all experienced in VR/AR/game development (M = 1.6, SD = 0.92). Non-technical students reported no experience at all in programming (M = 1.10, SD = 0.30) or VR/AR/game development (M = 1.00, SD = 0.00). MT students reported the highest experience with 3D modeling (M = 4.00, SD = 1.00), followed by CS students (M = 3.10, SD = 1.45). Non-technical students again reported no experience at all with 3D modeling (M = 1.10, SD = 0.30).

7.5. Results

We recorded the objective measures of TCT, TCR, perceived cognitive load (NASA rTLX) [45], and perceived usability (SUS) [46]. As this study is exploratory in nature, the objective measures of TCT and TCR are descriptively reported on the task level to show trends in the data, but inferentially analyzed and reported at sub-task level across the three tasks to have sufficient power for the statistical tests. The perceived cognitive load is also descriptively reported on the authoring-task level, but the average cognitive load across the experiment is analyzed using inferential statistics.

7.5.1. Task Completion Times

In terms of the Task-Completion Times (see Figure 10) of the first authoring task, CS students achieved the fastest average TCT of 3.11 min (SD = 3.39 min), followed by MT students with a TCT of 3.67 min (SD = 4.24 min). With a TCT of 4.86 min (SD = 3.84 min), non-technical students were on average the slowest during the first authoring task.

For the second authoring task, MT students achieved an average TCT of 1.33 min (SD = 1.01 min), closely followed by CS students with 1.47 min (SD = 1.03 min). Non-technical students on average needed 1.66 min (SD = 1.08 min) to complete the second authoring task. For the third authoring task, MT students were the fastest, where they on average needed 2.15 min (SD = 3.10 min) per sub-task. CS students achieved an average TCT of 2.21 min (SD = 2.74 min) and non-technical students an average of 2.6 min (SD = 3.45 min).

Not shown in Figure 10 are six outliers. Three outliers in the first authoring task are the first occurrences of the sub-task of the "action" category (see Figure 11): CS students on average needed 13.45 min, MT students 16.6 min, and non-technical students 15.8 min to solve this sub-task. The other three outliers not visible are part of the third authoring task and are the first occurrences of the sub-task category "fork-action". Here, CS students on average needed 9.80 min, MT students 11.96 min, and non-technical students 12.32 min to solve this sub-task.

Figure 11 shows the average task completion times for the occurrences of the sub-tasks in each sub-task-category, including the outliers. Notably, TCTs for all sub-tasks decreased not only consistently, but also to a similar degree in each of the student groups when occurring repeatedly.

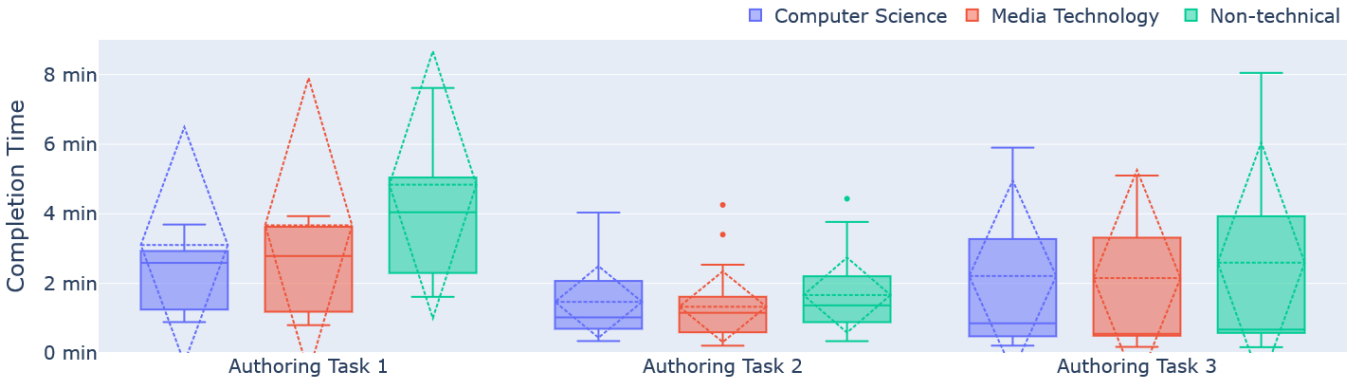

Figure 10. Average Task Completion Time (TCT) of each of the student groups (Computer Science, Media-Technology, and non-technical students) for each of the three authoring tasks the participants authored during the study.

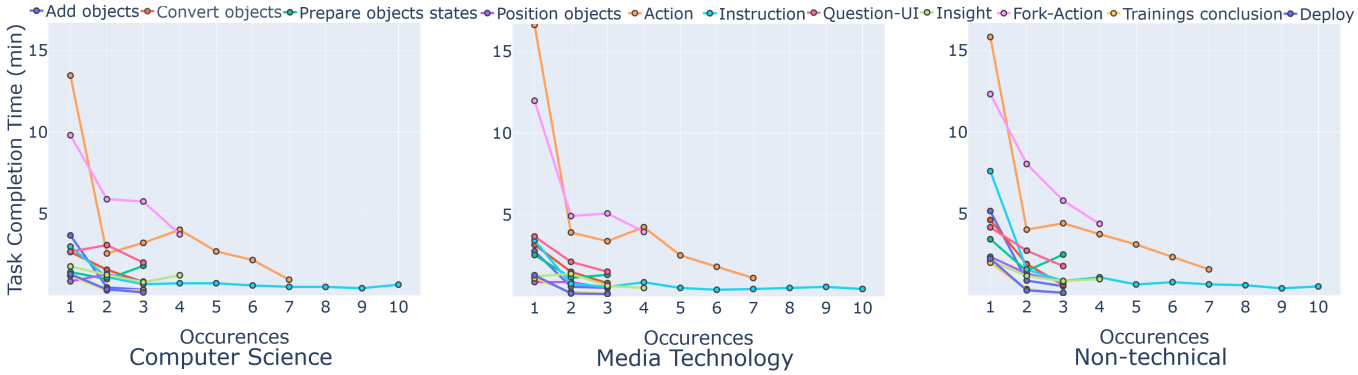

Figure 11. The average Task Completion Time (TCT) after each occurrence of one of the ten sub-task types for each of the student groups (Computer Science, Media-Technology, and non-technical students).

As the assumption of normality (Shapiro–Wilk test) was satisfied and Levene's test considered the populations' variance to be equal ($p = 0.646$), we conducted an ANOVA to check for differences of the average TCT across all 47 sub-tasks between the groups. The one-way ANOVA revealed no statistically significant differences in average TCT between CS students, MT students, or non-technical students ($F(2,27) = 2.79$, $p = 0.079$).

7.5.2. Task Completion Rates

Table 2 shows the average Task-Completion Rate (TCR) of the 47 sub-tasks across the three authoring tasks (11 for authoring task 1, 21 for 2, and 15 for 3) depending on the participant's group. Here, the reported TCR is split into four levels. On the first level, "no help", participants completed the task without any help or hints. On the second level, "documentation hint", participants were given a hint of where in the documentation the solution for their current task could be found. "Solution hint" was an explicit, predefined hint on how to solve the sub-task, which was shown to the participants when they were still not able to solve the task with the documentation hint. If this hint was also not sufficient, participants were helped by the experimenter to complete the sub-task ("explicit help").

While we do not have statistical power or sample size to deploy a two-way mixed-design analysis-of-variance model, there are some interesting descriptive trends that are apparent. For example, in the first authoring task, while in the CS and MT group, participants on average were able to complete over 80% of the sub-tasks without any help, this was only true for 64% of non-technical students. With an average percentage of completed sub-tasks without any help of 96% for the CS group, 98% for the MT group, and 92% for the non-technical students, this gap was narrowed with familiarity with the sub-tasks in the second authoring task. This is until non-linear action chains were introduced into the third authoring task, where the CS group and MT group both retained an average completion percentage of sub-tasks without help of above 90%, while the non-technical group reported the highest average percentage of sub-tasks only completed with explicit help by the experimenter (10%). This was mainly caused by the non-linear "Fork Actions", which the non-technical students struggled with. Here, the majority of them needed explicit help from the experimenter when it first occurred, while only one participant for the CS and MT students needed explicit help. Also notable is the fact that for the CS and MT students, the documentation hint was often sufficient (see the docu hint percentage >= solution hint percentage in Table 2), while for the non-technical student's solution hints were required more often (see the solution hint percentage > docu hint percentage in Table 2)

Table 2. The average Task Completion Rate (TCR) for each of the three authoring tasks, grouped by the participants' study program and reported on 4 levels: without any help, with a hint of where in the documentation the solution is described, with a predefined solution hint, or with explicit help.

Help/Hint	Computer Science	Media Technology	Non-Technical
	Authoring Task 1		
No Help	81.90% (SD = 23.00)	80.00% (SD = 18.00)	64.00% (SD = 26.00)
Docu Hint	9.90% (SD = 12.00)	11.00% (SD = 13.00)	12.00% (SD = 10.00)
Solution Hint	4.50% (SD = 8.70)	3.60% (SD = 4.60)	16.00% (SD = 15.00)
Explicit Help	3.60% (SD = 8.70)	5.40% (SD = 4.60)	8.10% (SD = 6.60)
	Authoring Task 2		
No Help	96.00% (SD = 7.80)	98.00% (SD = 0.00)	92.00% (SD = 9.00)
Docu Hint	1.50% (SD = 3.40)	1.00% (SD = 3.20)	1.00% (SD = 3.20)
Solution Hint	0.50% (SD = 1.60)	1.00% (SD = 3.20)	6.70% (SD = 5.90)
Explicit Help	1.90% (SD = 6.00)	0.00% (SD = 0.00)	0.50% (SD = 1.60)
	Authoring Task 3		
No Help	93.00% (SD = 9.70)	91.00% (SD = 9.50)	82.00% (SD = 10.00)
Docu Hint	2.00% (SD = 4.40)	3.40% (SD = 4.70)	1.40% (SD = 3.00)
Solution Hint	2.70% (SD = 4.60)	2.70% (SD = 4.60)	6.80% (SD = 6.30)
Explicit Help	2.70% (SD = 8.50)	2.70% (SD = 6.50)	10.00% (SD = 9.60)

Combining the task completion rates of all 47 sub-tasks and interpreting them as ranks ranging from one (completed without help) to four (explicit help of the experimenter), we can check for differences between groups with a non-parametric test. Therefore, a Kruskal–Wallis H test was used. It indicated a significant difference in the TCR between the groups, $\chi^2(2) = 7.63$, $p = 0.022$, with a mean rank score of 10.85 for CS students, 14.2 for MT students, and 21.45 for non-technical students. Dunn's posthoc test using a Bonferroni corrected alpha of 0.017 indicated that the mean rank of CS students and non-technical students was significantly different ($p = 0.007$). The differences between CS students and MT students ($p = 0.39$) and MT and non-technical students ($p = 0.065$) were not significantly different.

7.5.3. Perceived Cognitive Load

To measure the perceived cognitive workload of the participants, the non-weighted version of the NASA TLX questionnaire [45] (NASA rTLX) was used. Participants had to answer the questionnaire items on a 7-point Likert scale. The results were then normalized towards a 1–100 score. This is not the standard application method to evaluate the NASA rTLX but is commonly used this way in the literature to achieve scale consistency while already utilizing Likert scales [47,48]. Figure 12 shows the measured scores for each of the three tasks and participant groups. For the first authoring task, CS students reported the lowest perceived cognitive load with an average score of 31.67 (SD = 12.56), followed by MT students (M = 37.22, SD = 11.12), while non-technical students rated their perceived cognitive load the highest (M = 45.83, SD = 3.45). For authoring task two, CS students reported an average NASA rTLX score of 22.50 (SD = 6.51), MT students reported an average of 33.89 (SD = 10.67), and non-technical students an average of 34.44 (SD = 12.12). This trend continued for the final authoring task, where CS students again rated their perceived cognitive load the lowest (M = 29.77, SD = 12.30), followed by MT students (M = 35.56, SD = 7.11). Again, non-technical students rated their perceived cognitive load the highest (M = 42.24, SD = 16.89).

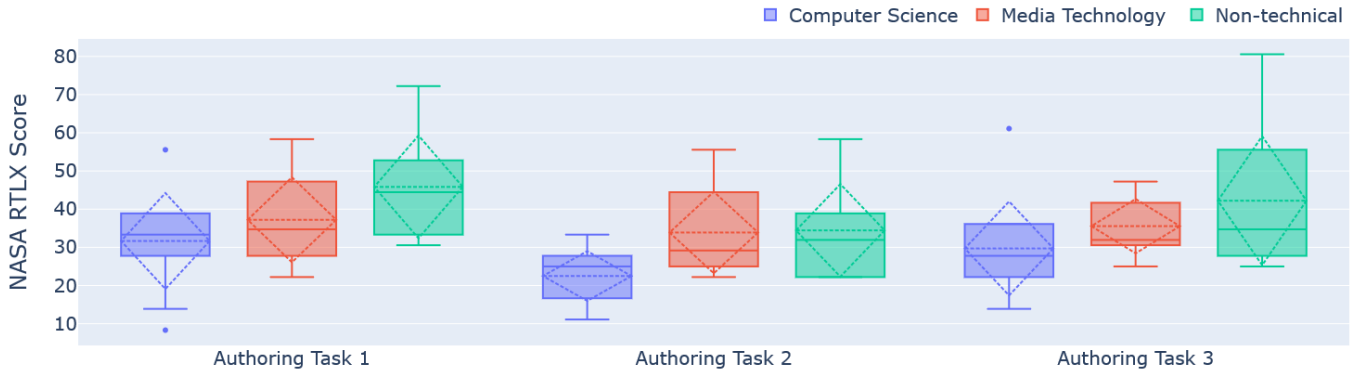

Figure 12. Average perceived cognitive load measures with the NASA rTLX [45] for each authoring task grouped by participants' study program, normalized to a 1–100 score.

As the assumption of normality was satisfied (Shapiro–Wilk test) and Leven's test considered the population's variance to be equal (p = 0.39), a one-way ANOVA was used on the average NASA rTLX score of all three measurement points after each authoring task. It revealed statistically significant differences between at least two groups ($F(2, 27) = 3.3924$, $p = 0.0485$). Tukey's HSD test for multiple comparisons indicated that CS students perceived significantly lower average cognitive load throughout the authoring process compared to non-technical students ($p = 0.0392$, C.I. [0.5534, 25.18]). No statistically significant differences were found between the MT and CS ($p = 0.29$) and MT and non-technical students ($p = 0.55$).

7.5.4. Perceived Usability

Analyzing the perceived usability, reported through the SUS questionnaire after completing all three authoring tasks, using the SUS Analysis Toolkit [44], CS students reported a SUS study score of 85.75 (SD = 9.88). This would be considered "Best Imaginable" usability according to Bangor et al. [36] and is above the non-empirical, but commonly used, industry benchmark of SUS study scores of 80 [37]. MT students reported a SUS study score of 79.25 (SD = 9.36), which would be considered "Good" usability [36]. The non-technical students reported a SUS study score of 60.25 (SD = 15.14). This would be considered a below-average, "OK" usability according to Bangor et al. [36] but would still be marginally acceptable usability [41]. A sample size of $n = 10$ for each of the groups should be 80% conclusive, according to Tullis et al. [38]. The results of the SUS are visualized in Figure 13.

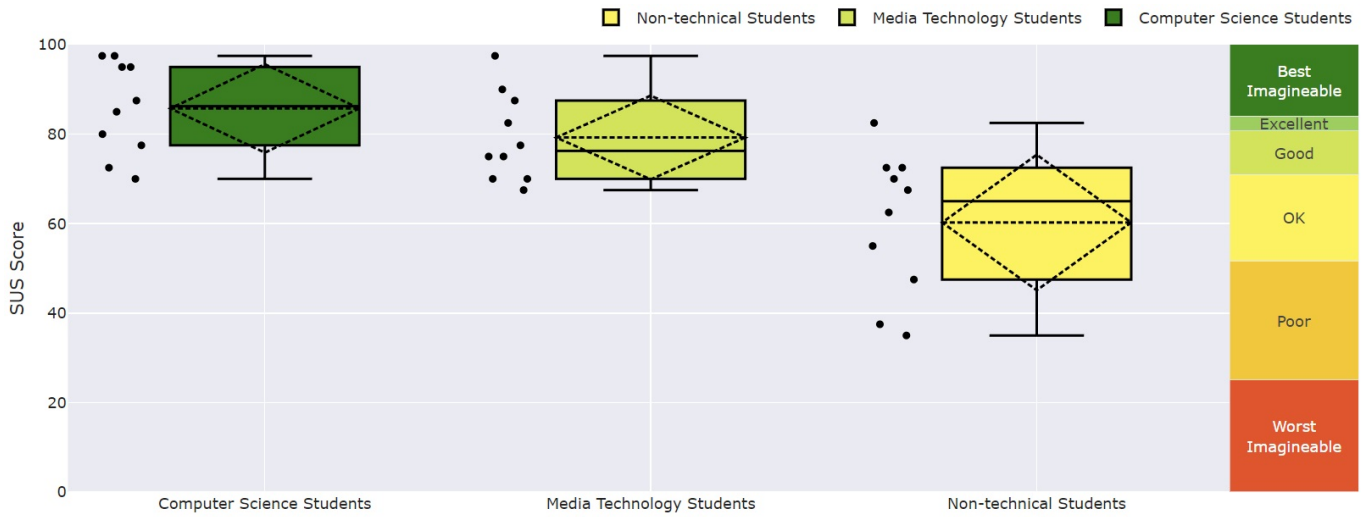

Figure 13. The perceived usability of the TrainAR authoring tool reported as SUS scores, grouped by the study program of the participants, plotted with the SUS Analysis Toolkit [44].

The assumption of normality was satisfied (Shapiro–Wilk test) and Levene's test considered the population's variance to be equal ($p = 0.293$). Therefore, a one-way ANOVA was conducted to compare the effect of the three groups on the perceived usability, reported through SUS scores. The one-way ANOVA revealed that there was a statistically significant difference in SUS scores between at least two groups ($F(2, 27) = 11.439$, $p = 0.00025$). Tukey's HSD test for multiple comparisons indicated that CS students reported significantly higher SUS scores compared to non-technical students ($p = 0.00025$, 95% C.I. = [11.7624, 39.24]), and MT students reported significantly higher SUS scores compared to non-technical students ($p = 0.00538$, 95% C.I. = [5.2624, 32.7376]). No statistically significant differences in SUS scores were found between CS students and MT students ($p = 0.479$).

7.5.5. Qualitative Feedback

To gather qualitative feedback for TrainAR, participants were asked what they liked and disliked while working with the authoring tool. When prompted to answer what the participants liked about TrainAR, eleven participants mentioned working with the visual scripting nodes and how these made it possible to implement complex training sequences without requiring any programming knowledge. Positively highlighted in particular was the visual representation of the processing logic. This was described as "intuitively understandable" and "clear". They stated that, once the basic concept was understood, it was "absolutely no problem applying them to the various tasks". Further, highlighted positively was the easy deployment process of a training to the handheld device. This made it possible to quickly identify and fix bugs.

When asked what the participants disliked about using TrainAR, three mentioned referencing the TrainAR Objects in the script nodes by their name. This was described as tedious and error-prone, because of typos, and participants wished that this could be performed by "drag and drop". Another three participants mentioned problems while moving objects in the 3D environment of the authoring tool (note: this was caused by a bug, where objects when dragged and dropped into the scene would sometimes not be placed at the correct height of the reference setup). Furthermore, the conversion process for TrainAR Objects was described as tedious by four of the participants, especially when multiple objects were involved, since the conversion had to be performed one-by-one and could not be performed in batches. Lastly, three participants from the MT and non-technical group mentioned that they may need help from a "technical person" to use TrainAR to its fullest potential.

When asked if they had any further remarks about TrainAR or the study, five participants from the MT and non-technical groups mentioned they had fun while using TrainAR.

Three participants in the CS and MT group answered they thought TrainAR is versatile in use and could see its potential. Two CS students stated that it provides an interesting first insight into AR development.

7.5.6. Self-Assessment: Independently Creating a TrainAR Training

Finally, we asked participants how much they would agree with the statement "I think I would be able to create AR trainings on my own using TrainAR" on a seven-point Likert scale. Both the CS students (M = 6, SD = 1.15) and the MT students (M = 5.8, SD = 1.03) agreed with the statement, while the non-technical students somewhat agreed (M = 4.8, SD = 1.87) with it. As the sample size was small for Likert-scale data, and the normality assumption was violated, a Kruskal–Wallis H test was performed, which indicated that there was a non-significant difference in the dependent variable between the three groups, $\chi^2(2) = 2.64$, $p = 0.267$, with a mean rank score of 18.05 for CS students, 16.4 for MT students, 12.05 for the non-technical students. The effect size was small ($\eta^2 = 0.024$).

When asked to provide the reasoning behind their self-assessment, seven participants (all either MT or CS students) stated working with TrainAR, after a short familiarization phase, is easy to understand and use. Positively highlighted here were the visual state machine nodes, which were described as "intuitive" and "explained in an understandable way". Furthermore, three participants mentioned that the documentation was an enormous help in understanding what each element of TrainAR, especially the state machine nodes, does and made it possible to get them started quickly. Three participants, however, mentioned that they would not know how to acquire the 3D models necessary for creating trainings when they are not provided as they were during the study. One of the participants described the reasoning behind their self-assessment as "still too many questions" and another stated that they would need help from an expert to guide them if they had to create a training with TrainAR.

8. Discussion

The authoring tool of the TrainAR framework allows domain experts to create their own procedural AR trainings by utilizing the evaluated TrainAR interaction concept, including its onboarding and technical utility, allowing authors to focus on the content of the training, and not having to worry about the technical aspects or AR-specific implementation challenges. Authors, if they choose to do so, can follow our didactic consideration framework but can also implement their own didactic ideas. With TrainAR being completely free, published as open-source under the MIT license, being fully documented, and using handheld AR devices as the target hardware, the created procedural AR trainings are realistically scalable today. This enables bring-your-own-device methodologies, self-directed learning, location-independent learning, and self-paced preparation or retention opportunities through interactive AR trainings, which are engaging and incorporate psychomotor learning components [39]. This pragmatic and, more importantly, holistic approach to the authoring of interactive AR trainings is novel in the literature and should not only help the efficiency of the authoring process but also provide guidance for the created trainings to follow proven principles. We think that this will also help to resolve the causality dilemma of not knowing what an AR training could look like, before even starting to author them, but also not being able to develop an AR training before having an extensive process description of the training.

As discussed in Section 4.1, we deliberately chose to implement the TrainAR authoring tool as a Unity extension and developed the TrainAR Statemachine and UI layout options to be extendable and replaceable. Ultimately, programmers can utilize the full Unity functionality seamlessly with the framework. Where the TrainAR scope and documentation ends, there are a good number of documentation, tutorials, and getting-started utilities on how to use the Unity engine. In this context, TrainAR should enable and accelerate development, but never be a limiting factor. While the authoring tool does not require programming skills and is designed as a low-level content-design framework, we are aware

of the inherent trade-off of this introducing interface complexity, compared to standalone authoring tool approaches but believe that this trade-off is merited.

This pragmatic, open perspective continues in the didactic consideration aspects of the framework. From our first explorations into different contexts and the work of the partners on their TrainAR trainings, we expect the utility (see Figure 7) and therefore inherently also the didactic perspective on the utilization of the TrainAR trainings to differ significantly based on the context-specific needs. Therefore, while we provide didactic considerations [17], they are more of a "didactic cookbook" for authors to use according to their own tastes than strict rules we envision for the framework to follow.

In its current stage, the gathering or generation of 3D assets to use in the training likely remains the biggest technical challenge. Additionally, this aspect was also excluded during our usability evaluation of the authoring tool. For one, there are multiple ways to obtain assets, ranging from online asset stores, over free online databases, to 3D scanning, making it challenging to actually evaluate this aspect systematically. However, more importantly, we believe that the asset generation will be increasingly simplified with technological advancements, as already today, modern smartphones can create acceptable 3D assets through LiDAR scanning, and zero-shot 3D asset generation through AI (Artificial Intelligence) is showing promising recent scientific results [33]. We believe that, eventually, the 3D asset generation, and therefore the TrainAR Object aspect, will become decreasingly challenging, and we therefore explicitly designed it with this development in mind.

8.1. Evaluations of TrainAR Trainings

As the current and forthcoming results of the utility and usability of TrainAR trainings look promising, this is at least a first indication that the TrainAR framework is providing the necessary utility to create TrainAR trainings that elicit the desired learning outcomes, are usable, and enjoyable. Though this is promising, and distinctive application contexts were chosen deliberately, the generalizability of these results has to be confirmed through the application of the framework to novel contexts. Notably, while the feedback of other researchers, utilizing TrainAR for more than two years at the time of publishing this paper, indicates that the provided utility is largely sufficient, some implementations of TrainAR already required context-specific extensions, such as logbooks and interactive chemical graphs in the MARLab training application [40]. Moreover, while we were able to achieve consistently high perceived usability, this can only provide the upper limit and show that it is possible to create usable trainings but does not guarantee that usage of the TrainAR authoring tool will inherently lead to trainings with good usability.

The fact that the TrainAR authoring tool can be used effectively to create procedural and interactive handheld AR trainings is shown through our use and the use of partner universities that utilized TrainAR to create interactive procedural trainings and learning games for smartphones and tablets. These trainings span across a wide range of topics, targeted media competencies, and ages, with consistent observations, especially in terms of the enjoyability of the trainings, which previous work identified as the most important factor for usage intention [5].

8.2. Evaluation of the TrainAR Authoring Tool

The systematic usability evaluation of the TrainAR authoring tool revealed several interesting insights. While there were trends in the TCTs across all sub-tasks but also on the task level, no statistically significant differences were found. When categorizing the sub-task by the task category, a clear trend can be observed, where sub-tasks that occur multiple times are subsequently completed faster. This decline in subsequent TCT is especially steep after the first occurrence of a sub-task category, and is seemingly quite consistent across the groups (see Figure 11). This might be an indication that people can learn the handling of the authoring tool quickly, and there is only an initial hurdle to conceptual understanding. Interesting here are also the two outliers of the first occurrence of the action node and the first occurrence of the fork-action node, which require by far the most time to complete

when they are introduced, but this decreases sharply. While we expected as much and already provided getting-started guides, we interpret this as a call for even more in-depth onboarding materials that provide the author with assistance on this initial hurdle, e.g., in more digestible formats such as video tutorials or practical course materials.

We tried to increase the difficulties of each of the three authoring tasks by first using a linear task of actions in the first, introducing quizzes and custom actions into the second, and then introducing non-linear flows in the third task. The TCT by task indicates that the complexity increment of quizzes was not nearly as challenging as the introduction of non-linear flows, as there is a decline in TCT between the first and second authoring task, but then, while the median stays consistent, a noticeable increase in the average and standard deviation of the TCT for the third authoring task, as can be seen in Figure 10. This was caused by fork actions. Looking at these and contextualizing them with the TCR, this was clearly caused by the introduction of non-linear flows across the groups.

Generally, significant differences were found for the TCR between the groups. Here, CS students performed significantly better than non-technical students, and the difference between MT and non-technical students was approaching significance ($p = 0.065$). An increase in TCR from the first to the second authoring task is consistent with the TCT and across the groups. Notably, non-technical students had very noticeable problems as soon as the non-linear fork-actions were introduced, which, in contrast to TCT, was not as apparent in the other groups. This is likely due to familiarity with programming and node-based systems in asset-creation pipelines that the MT and CS groups brought with them, and should be considered for the documentation and onboarding material.

This trend continued for the perceived cognitive load, where CS students reported significantly lower scores than the non-technical students, but no other differences were found. Again, in line with TCT and TCR, within the group, the perceived cognitive load first decreased after the second task but then for all groups increased with the introduction of the fork-actions again.

Most importantly, the perceived usability reported through the SUS was the self-reported measure of usability in our experiment. Here, significant differences were found between CS and non-technical and MT and non-technical students, but not between the CS and MT groups. As this is in line with the objective measures of TCT and TCR and somewhat in line with the perceived cognitive load, we believe this supports the hypothesis that the tool is usable by media technologists and domain experts with high media competency in its current state. The results indicate that domain experts with lower levels of media competency might struggle with the current state of the authoring tool, with the most difficult challenges being asset acquisition and non-linear stateflows. Nonetheless, it might be possible to lower the entrance barrier through documentation and getting-started guides, e.g., in more approachable formats such as videos or practical course material. Additionally, when asked for their self-assessment if they think they were able to create a TrainAR training, even the non-technical students somewhat agreed, which did not significantly differ from the other groups, which agreed.

While we think these results are promising, they are limited in several ways. For one, recruiting students from three groups with implied differences is not the ideal experimental setup, as stated in Section 7.2, but was chosen for practical reasons. Notably, at least the self-reported measures were in line with the group expectations. Additionally, the sample size was small, and we were aware that the prospective statistical power would be low, but larger sample sizes were not realistic, as the study took roughly two to three hours to conduct per participant and one-on-one study support was needed. Nonetheless, our results, and the trends that can be shown descriptively and through the qualitative feedback, are in line with our expectations. Importantly, they are also in line with the observations we made when sharing TrainAR with other researchers over the last two years, and using the authoring tool iteratively in several practical tutorial sessions at our university. We are therefore satisfied with these results and will address the observations in subsequent documentation and course material and move on to the next evaluation stage.

8.3. In-The-Wild Testing Approach & Ongoing Evaluations

Realistically, there is no lab-study or prototypical evaluation that can fully answer our questions. Gathering requirements before having an initial implementation of TrainAR trainings and an authoring tool to create them still faces the causality dilemma of the missing shared understanding of what procedural task trainings would even look like for Handheld AR, as discussed in Section 3. Additionally, it is challenging to distinguish general needs for the framework from context-specific needs. We are cautiously optimistic about the provided set of utility, based on our non-representative observations in our contexts, the feedback by other researchers, and the systematic usability evaluation of the authoring tool. Still, generalizable insights will only be gathered after open-source publication of the authoring tool and its application by independent parties.

With this publication of the entire TrainAR framework, the evaluation results, and open sourcing of the authoring tool, we are therefore starting an evaluation period in line with the field usability testing methodology [49], and other researchers and educators are encouraged to deploy the tool and provide feedback (e.g., through email or GitHub Issues) from their real usage experience for additional refinements or future directions of TrainAR. Additionally, we will use the TrainAR framework and apply it holistically in our next evaluation iteration. Here, we will let teachers author TrainAR trainings with the tool and not only evaluate the authoring process itself but also then let them use the created trainings in a classroom setting as a multimedia learning intervention, where we will then gather feedback on the utility and usability of the training created by them, not us. Furthermore, we will continue to deploy the authoring tool into the practical parts of an apprenticeship course again. This will allow us to iteratively evaluate further and improve the usability of the authoring tool, while providing first insights into AR development for the students.

9. Conclusions

In this paper, TrainAR's framework components were discussed from a technical perspective, and a visual scripting-based authoring tool was proposed for the TrainAR framework: The TrainAR authoring tool. The authoring tool has been published as an open-source project (https://github.com/jblattgerste/TrainAR, accessed: 31 March 2023), containing the full source code for the TrainAR interaction concept and authoring functionality as a Unity extension under the MIT license. In addition to a full documentation (https://jblattgerste.github.io/TrainAR/, accessed: 31 March 2023), which is included with the authoring tool, getting-started guides, and tutorials are available and several TrainAR scenarios are already developed, evaluated, and documented in reference videos. They showed good usability, were enjoyable by the trainees, and showed utility results such as increased motivation. A systematic usability evaluation of the TrainAR authoring tool revealed that, while all groups were able to use the authoring tool and improved over time during the study, the likely target group of the authoring tool is media technologists or domain experts with high media competency. This is mainly due to concepts such as the formalization of non-linear flows of states, which are hard to grasp for people with lower levels of media competency, and the need to acquire appropriate 3D assets. Nonetheless, programming or expertise in software engineering does not seem to be required to utilize the authoring tool, and we believe that our current results suggest that the TrainAR authoring tool is a useful contribution to the current state of the AR authoring landscape.

Current & Future Work

Besides our efforts to continuously evaluate and iteratively improve the state of the authoring tool's source code for the current scope of the TrainAR framework as stated in Section 8.3, we also plan to expand TrainARs ideas and perspectives in several directions and hope for future cooperations and third-party contributions to our open-source project.

One important technical aspect we are trying to address currently is training distribution aspects. Here, we are exploring if we could improve the deployment and distribution

aspect of authored TrainAR trainings for users during the development phase and formative testing and the actual deployment afterward, as currently, when authors create trainings, they would have to either manually distribute the training as compiled apps for Android or iOS or would have to publish them into an asset store, which, in our experience, is a daunting endeavor.

Furthermore, we are currently working on a higher-level onboarding utility for the TrainAR Authoring Tool. While the extensive documentation already includes getting-started guides and examples to utilize, we are currently working on video material in the "tutorial" format, which are more specifically targeted at potential users that are neither programmers nor media technologists, in the hopes to lower the barrier of entrance without limiting the expandability of TrainARs authoring tool.

Additionally, we plan to work on a didactic white paper that specifies more clearly what we believe specific components of the framework can achieve and how we envision them to be used (e.g., feedback nodes, insights, instructions, or quiz elements). While this is discussed in detail in our paper on the didactic framework [17] and summarized in the technical documentation, we want to make this information more approachable and provide it in formats in line with domain experts' expectations.

We are also interested in expanding TrainAR towards incorporating physical object and marker tracking, to not only use virtual objects but also integrate physical objects into the flow during the training. While this was explored during development and is not particularly challenging from a technical perspective, as tracking libraries for this already exist, considerations must be made for the didactic concept and how to integrate the physical material seamlessly into the framework, as this might impact not only the usability but also the current strengths of TrainARs' ideas of location-independent learning, material/cost-savings, and immediate scalability of authored AR trainings [17].

Finally, while interaction concepts that can be used for procedural trainings such as the MRTK exist for AR HMDs, we would like to explore if expanding TrainAR towards including HMDs as a target platform would be feasible and viable, as not only the visual representation of the stateflow but also some didactic considerations and modules could significantly add to the state of AR authoring tools in the HMD-based training context.

Author Contributions: Conceptualization, J.B. (Jonas Blattgerste); software, J.B. (Jonas Blattgerste) and J.B. (Jan Behrends); validation, J.B. (Jonas Blattgerste), J.B. (Jan Behrends) and T.P.; writing—original draft preparation, J.B. (Jonas Blattgerste) and J.B. (Jan Behrends); writing—review and editing, J.B. (Jonas Blattgerste) and T.P.; visualization, J.B. (Jonas Blattgerste); supervision, T.P.; project administration, J.B. (Jonas Blattgerste); funding acquisition, T.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by grant 16DHB3021, project "HebAR—AR-Based-Training-Technology", by the German Ministry for Education and Research (BMBF).

Data Availability Statement: The data for the evaluation reported in Section 7 is not publicly available due to privacy concerns. Participants consented to anonymized, but not pseudonymised, publication.

Acknowledgments: Besides the authors of this paper, Sven Janßen contributed to the technical development of the TrainAR Interaction Concept, Authoring Tool, and TrainAR training (1) featured in Section 6. Furthermore, Nils Münke contributed to the development of the TrainAR training (1) and Jannik Franssen to the development of TrainAR training (2). Additionally, as part of project CHARMING, training (3) was developed by Jessica Lizeth Domínguez Alfaro and Stefanie Gantois at KU Leuven and training (4) and (5) by Michaela Arztmann and Jeroen Hijzelendoorn at Utrecht University. We furthermore thank the students who developed TrainAR trainings during our courses and lectures and provided valuable feedback.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AR	Augmented Reality
VR	Virtual Reality
MR	Mixed Reality
HMD	Head-mounted Device
UI	User Interface
SUS	System Usability Scale
TCT	Task Completion Time
TCR	Task Completion Rate
CS	Computer Science
MT	Media Technology
MRTK	Mixed Reality Toolkit

References

1. Santos, M.E.C.; Taketomi, T.; Yamamoto, G.; Rodrigo, M.M.T.; Sandor, C.; Kato, H. Augmented reality as multimedia: The case for situated vocabulary learning. *Res. Pract. Technol. Enhanc. Learn.* **2016**, *11*, 1–23. [CrossRef] [PubMed]
2. Ozdemir, M.; Sahin, C.; Arcagok, S.; Demir, M.K. The effect of augmented reality applications in the learning process: A meta-analysis study. *Eurasian J. Educ. Res.* **2018**, *18*, 165–186. [CrossRef]
3. da Silva, M.M.; Teixeira, J.M.X.; Cavalcante, P.S.; Teichrieb, V. Perspectives on how to evaluate augmented reality technology tools for education: A systematic review. *J. Braz. Comput. Soc.* **2019**, *25*, 1–18. [CrossRef]
4. Quintero, J.; Baldiris, S.; Rubira, R.; Cerón, J.; Velez, G. Augmented reality in educational inclusion. A systematic review on the last decade. *Front. Psychol.* **2019**, *10*, 1835. [CrossRef] [PubMed]
5. Wojciechowski, R.; Cellary, W. Evaluation of learners' attitude toward learning in ARIES augmented reality environments. *Comput. Educ.* **2013**, *68*, 570–585. [CrossRef]
6. Buabeng-Andoh, C. Factors influencing teachersâ adoption and integration of information and communication technology into teaching: A review of the literature. *Int. J. Educ. Dev. Using Inf. Commun. Technol. (IJEDICT)* **2012**, *8*, 136–155.
7. Wu, H.K.; Lee, S.W.Y.; Chang, H.Y.; Liang, J.C. Current status, opportunities and challenges of augmented reality in education. *Comput. Educ.* **2013**, *62*, 41–49. [CrossRef]
8. Tzima, S.; Styliaras, G.; Bassounas, A. Augmented reality applications in education: Teachers point of view. *Educ. Sci.* **2019**, *9*, 99. [CrossRef]
9. Saforrudin, N.; Zaman, H.B.; Ahmad, A. Technical skills in developing augmented reality application: Teachers' readiness. In Proceedings of the International Visual Informatics Conference, Selangor, Malaysia, 9–11 November 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 360–370. [CrossRef]
10. Cubillo, J.; Martin, S.; Castro, M.; Boticki, I. Preparing augmented reality learning content should be easy: UNED ARLE—An authoring tool for augmented reality learning environments. *Comput. Appl. Eng. Educ.* **2015**, *23*, 778–789. [CrossRef]
11. Schmalstieg, D.; Langlotz, T.; Billinghurst, M. Augmented Reality 2.0. In *Virtual Realities*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 13–37. [CrossRef]
12. Hampshire, A.; Seichter, H.; Grasset, R.; Billinghurst, M. Augmented reality authoring: Generic context from programmer to designer. In Proceedings of the 18th Australia Conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments, New York, NY, USA, 20–24 November 2006; pp. 409–412. [CrossRef]
13. Nebeling, M.; Speicher, M. The trouble with augmented reality/virtual reality authoring tools. In Proceedings of the 2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Munich, Germany, 16–20 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 333–337. [CrossRef]
14. Ashtari, N.; Bunt, A.; McGrenere, J.; Nebeling, M.; Chilana, P.K. Creating augmented and virtual reality applications: Current practices, challenges, and opportunities. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, New York, NY, USA, 25–30 April 2020; pp. 1–13. [CrossRef]
15. Nebeling, M. Playing the Tricky Game of Toolkits Research. In Proceedings of the Workshop on HCI. Tools at CHI, Denver, CO, USA, 6–11 May 2017.
16. Kiesler, N.; Schiffner, D. On the lack of recognition of software artifacts and it infrastructure in educational technology research. In *20. Fachtagung Bildungstechnologien (DELFI)*; Gesellschaft für Informatik e.V.: Bonn, Germany, 2022; pp. 201–206. [CrossRef]
17. Blattgerste, J.; Luksch, K.; Lewa, C.; Pfeiffer, T. TrainAR: A Scalable Interaction Concept and Didactic Framework for Procedural Trainings Using Handheld Augmented Reality. *Multimodal Technol. Interact.* **2021**, *5*, 30. [CrossRef]
18. Dünser, A.; Walker, L.; Horner, H.; Bentall, D. Creating interactive physics education books with augmented reality. In Proceedings of the 24th Australian Computer-Human Interaction Conference, Melbourne, Australia, 26–30 November 2012; pp. 107–114. [CrossRef]
19. Liarokapis, F.; Mourkoussis, N.; White, M.; Darcy, J.; Sifniotis, M.; Petridis, P.; Basu, A.; Lister, P.F. Web3D and augmented reality to support engineering education. *World Trans. Eng. Technol. Educ.* **2004**, *3*, 11–14.

20. Laine, T.H.; Nygren, E.; Dirin, A.; Suk, H.J. Science Spots AR: A platform for science learning games with augmented reality. *Educ. Technol. Res. Dev.* **2016**, *64*, 507–531. [CrossRef]
21. Blattgerste, J.; Renner, P.; Pfeiffer, T. Authorable augmented reality instructions for assistance and training in work environments. In Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia, Pisa, Italy, 26–29 November 2019; pp. 1–11. [CrossRef]
22. Blattgerste, J.; Pfeiffer, T. Promptly authored Augmented Reality instructions can be sufficient to enable cognitively impaired workers. In Proceedings of the GI VR/AR Workshop. Gesellschaft für Informatik eV, online, 24–25 September 2020. [CrossRef]
23. Escobedo, L.; Tentori, M.; Quintana, E.; Favela, J.; Garcia-Rosas, D. Using augmented reality to help children with autism stay focused. *IEEE Pervasive Comput.* **2014**, *13*, 38–46. [CrossRef]
24. Lytridis, C.; Tsinakos, A.; Kazanidis, I. ARTutor—An augmented reality platform for interactive distance learning. *Educ. Sci.* **2018**, *8*, 6. [CrossRef]
25. Maloney, J.; Resnick, M.; Rusk, N.; Silverman, B.; Eastmond, E. The scratch programming language and environment. *ACM Trans. Comput. Educ. (TOCE)* **2010**, *10*, 1–15. [CrossRef]
26. Kelly, A.; Shapiro, R.B.; de Halleux, J.; Ball, T. ARcadia: A rapid prototyping platform for real-time tangible interfaces. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, Montreal, QC, Canada, 21–26 April 2018; pp. 1–8. [CrossRef]
27. Not, E.; Petrelli, D. Empowering cultural heritage professionals with tools for authoring and deploying personalised visitor experiences. *User Model. User-Adapt. Interact.* **2019**, *29*, 67–120. [CrossRef]
28. Mota, J.M.; Ruiz-Rube, I.; Dodero, J.M.; Arnedillo-Sánchez, I. Augmented reality mobile app development for all. *Comput. Electr. Eng.* **2018**, *65*, 250–260. [CrossRef]
29. Apaza-Yllachura, Y.; Paz-Valderrama, A.; Corrales-Delgado, C. SimpleAR: Augmented Reality high-level content design framework using visual programming. In Proceedings of the 2019 38th International Conference of the Chilean Computer Science Society (SCCC), Concepcion, Chile, 4–9 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–7. [CrossRef]
30. Nguyen, V.T.; Jung, K.; Dang, T. BlocklyAR: A Visual Programming Interface for Creating Augmented Reality Experiences. *Electronics* **2020**, *9*, 1205. [CrossRef]
31. Castillo, R.I.B.; Sánchez, V.G.C.; Villegas, O.O.V.; Lozoya, A.L. Node based visual editor for mobile augmented reality. *Int. J. Comb. Optim. Probl. Inform.* **2016**, *7*, 35–48.
32. Groenendyk, M. Cataloging the 3D web: The availability of educational 3D models on the internet. *Libr. Hi Tech.* **2016**, *34*, 239–258. [CrossRef]
33. Jain, A.; Mildenhall, B.; Barron, J.T.; Abbeel, P.; Poole, B. Zero-shot text-guided object generation with dream fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 867–876.
34. Technologies, U. Unity Visual Scripting Package. Available online: https://docs.unity3d.com/Packages/com.unity.visualscripting@1.8/manual/index.html (accessed on 13 February 2023).
35. Blattgerste, J.; Luksch, K.; Lewa, C.; Kunzendorf, M.; Bauer, N.H.; Bernloehr, A.; Joswig, M.; Schäfer, T.; Pfeiffer, T. Project Heb@ AR: Exploring handheld Augmented Reality training to supplement academic midwifery education. In *DELFI 2020–Die 18. Fachtagung Bildungstechnologien der Gesellschaft für Informatik eV*; Gesellschaft für Informatik e.V.: Bonn, Germany, 2020; pp. 103–108.
36. Bangor, A.; Kortum, P.; Miller, J. Determining what individual SUS scores mean: Adding an adjective rating scale. *J. Usability Stud.* **2009**, *4*, 114–123.
37. Lewis, J.R.; Sauro, J. Item benchmarks for the system usability scale. *J. Usability Stud.* **2018**, *13*, 158–167.
38. Tullis, T.; Stetson, J. A Comparison of Questionnaires for Assessing Website Usability. In Proceedings of the Usability Professional Association Conference 1, Minneapolis, MN, USA, 7–11 June 2004.
39. Blattgerste, J.; Franssen, J.; Arztmann, M.; Pfeiffer, T. Motivational benefits and usability of a handheld Augmented Reality game for anatomy learning. In Proceedings of the 2022 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), online, 12–14 December 2022; pp. 266–274. [CrossRef]
40. Domínguez Alfaro, J.L.; Gantois, S.; Blattgerste, J.; De Croon, R.; Verbert, K.; Pfeiffer, T.; Van Puyvelde, P. Mobile Augmented Reality Laboratory for Learning Acid–Base Titration. *J. Chem. Educ.* **2022**, *99*, 531–537. [CrossRef]
41. Bangor, A.; Kortum, P.T.; Miller, J.T. An empirical evaluation of the system usability scale. *Int. J. Hum.-Interact.* **2008**, *24*, 574–594. [CrossRef]
42. Arztmann, M.; Domínguez Alfaro, J.L.; Blattgerste, J.; Jeuring, J.A. Marie's ChemLab: A Mobile Augmented Reality Game to Teach Basic Chemistry to Children and Van Puyvelde, Peter. In Proceedings of the ECGBL 2022, Lusófona, Portugal, 6–7 October 2022; Academic Conferences and Publishing International: New York, NY, USA, 2022; pp. 65–72.
43. Putnam, C.; Puthenmadom, M.; Cuerdo, M.A.; Wang, W.; Paul, N. Adaptation of the System Usability Scale for User Testing with Children. In Proceedings of the Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, New York, NY, USA, 25–30 April 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1–7. [CrossRef]
44. Blattgerste, J.; Behrends, J.; Pfeiffer, T. A Web-Based Analysis Toolkit for the System Usability Scale. In Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments, Corfu, Greece, 29 June–1 July 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 237–246. [CrossRef]

45. Hart, S.G.; Staveland, L.E. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in Psychology*; Elsevier: Amsterdam, The Netherlands, 1988; Volume 52, pp. 139–183. [CrossRef]
46. Brooke, J. SUS: A quick and dirty usability scale. *Usability Eval. Ind.* **1995**, *189*, 4–7. [CrossRef]
47. Salvador, G.F.D.; Bota, P.J.; Vinayagamoorthy, V.; Plácido da Silva, H.; Fred, A. Smartphone-Based Content Annotation for Ground Truth Collection in Affective Computing. In Proceedings of the ACM International Conference on Interactive Media Experiences, Virtual, 21–23 June 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 199–204. [CrossRef]
48. Corazza, F.; Snijders, D.; Arpone, M.; Stritoni, V.; Martinolli, F.; Daverio, M.; Losi, M.G.; Soldi, L.; Tesauri, F.; Da Dalt, L.; et al. Development and Usability of a Novel Interactive Tablet App (PediAppRREST) to Support the Management of Pediatric Cardiac Arrest: Pilot High-Fidelity Simulation-Based Study. *JMIR Mhealth Uhealth* **2020**, *8*, e19070. [CrossRef]
49. Dumas, J.S.; Dumas, J.S.; Redish, J. *A practical Guide to Usability Testing*; Intellect Books: Bristol, UK, 1999.