Tutorial Generation For Virtual Reality from Example Playtroughs

Saeed Safikhani *, Dorian Lux, Dieter Schmalstieg, Johanna Pirker

Posted Date: 28 October 2024

doi: 10.20944/preprints202410.2102.v1

Keywords: Virtual Reality; Tutorial; Onboarding; Automation; User study

 doi:10.20944/preprints202410.2102.v1

Article

Tutorial Generation For Virtual Reality from Example Playtroughs

Saeed Safikhani [1,*], Dorian Lux [1], Dieter Schmalstieg [1,2] and Johanna Pirker [1,3]

[1] Graz University of Technology, Austria
[2] Universität Stuttgart, Germany
[3] Ludwig Maximilians Universität München, Germany
* Correspondence: s.safikhani@tugraz.at

Abstract: As virtual reality (VR) games appeal to increasingly large audiences, many newcomers need convenient and flexible onboarding sessions. Traditional methods of creating tutorial levels can be time-consuming and require additional effort from the development team. In this paper, we present a toolkit for computer-assisted tutorial generation in VR. We conducted an interview with developers and a between-subjects user study to evaluate our toolkit compared to a manually designed onboarding experience by measuring user performance in actual gameplay. The results indicate that our toolkit performs equivalently to manual tutorials in terms of user success. Most users subjectively prefer a manually designed tutorial primarily for its aesthetic value. These results are encouraging, as they prove the feasibility of semi-automated onboarding.

Keywords: virtual reality; tutorial; onboarding; automation; user study

1. Introduction

User training and education are important applications of virtual reality (VR) for various scenarios in schools, universities, and industry. Providing repeatable, safe and affordable learning environments helps users experience conditions that were previously difficult, dangerous, or costly [1]. A good example is the training of medical doctors, who must make critical decisions in a short period of time [2]. Repeating these conditions as an exercise in a safe VR simulation environment can train doctors to make the appropriate decisions in real-world situations. In addition to training scenarios, VR can also be used as a therapeutic and psychiatric intervention tool [3]. These use cases, on the one hand, and the increasing availability of VR devices, on the other hand, attract more and more users. New users need accessible, personalized and flexible onboarding sessions to start VR applications [4]. Onboarding helps them reduce the extensive cognitive load of using the new technology [5,6]. Cognitive load and satisfaction have been shown to be related [7]. Consequently, we expect user satisfaction and learning rates to increase as the cognitive load is reduced. A progressive onboarding session in the form of a tutorial can help users improve their experience [8].

Compared to traditional desktop games, the design of VR tutorials may need additional effort [3], as users may be less familiar with handling equipment and interacting with the virtual world. We believe that generating a tutorial from actual gameplay can help in recognizing such demanding circumstances. It can also help maintain the consistency of the VR experience and therefore support an uninterrupted sense of presence [9].

In this paper, we design, develop, and evaluate a toolkit for *generating VR tutorials from sample gameplay*. Our toolkit parses a gameplay sequence for the events required to achieve the game objective and distills them into a tutorial presentation. Since our agent replays variants of a playtrace (recording of game events), we do not require any form of artificial intelligence. Consequently, no access to the game code is made, and our framework is independent of the game's implementation. This approach stands in contrast to previous work, where event detection often relies on information extracted from game code, on an AI that can play the game, or both [10–14]. The method proposed in this study avoids the need for code analysis or artificial intelligence by extracting gameplay information from observing a human player playing the game in a sandboxed runtime system. Assuming that a

 doi:10.20944/preprints202410.2102.v1

competent human can be trusted to give a comprehensive and useful demonstration, our approach upgrades the information collected from a pure “ghost replay” into an interactive tutorial without requiring the effort required for manual content creation.

To evaluate our approach, we conducted an interview with game developers and a between-subjects user study. We compared our semi-automatically generated tutorials to manual ones using a "challenge" game level; see Figure 1. In addition, we report on interviews with three developers about the subjective quality of the tutorials generated with ATGVR. Based on the results of the user study, we compared the success rate, game duration, user experience, and user preferences between the two conditions. The result of this comparison can help us answer the following research questions:
RQ1: Is a semi-automatically generated tutorial equivalent to a manual tutorial in terms of user performance (required duration to solve the puzzle)?
RQ2: Is there a significant difference between the success rate of users who work with semi-automatic tutorials and those who work with manual tutorials?
RQ3: Is there a significant difference between users working with semi-automatic tutorials and users working with manual tutorials in regard to user experience (ease of use, presence, task demand)?

Answering these research questions can clarify the impact of the semi-automatic generated tutorial on user performance and give insight into user preferences to improve this toolkit in future work.

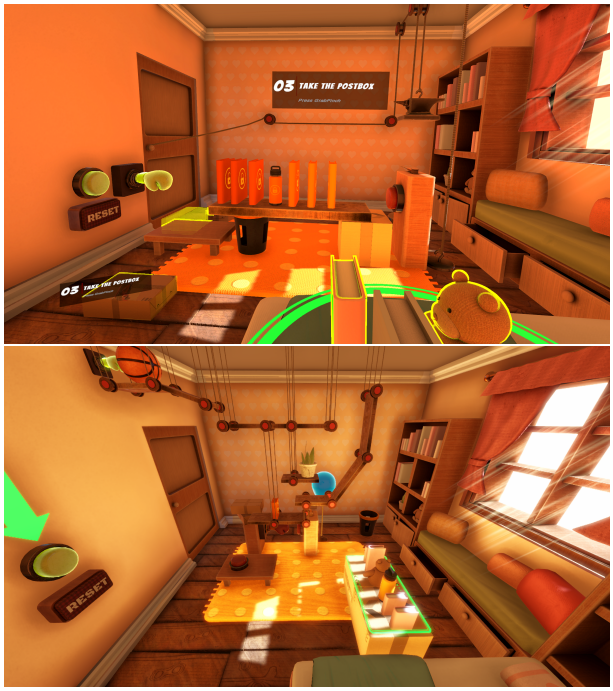

Figure 1. The structure of the (left) onboarding level, (right) challenge level. The game is a puzzle game involving dominoes.

2. Related Work

The term “tutorial” refers to the strategies employed by game developers to convey gameplay mechanics to players. Tutorials in games have a long tradition in the gaming industry, and users encounter tutorials in the early parts of every game [15]. Although researchers have studied the impact of tutorial presentation on user experience and performance [16–21], this area has been less explored in the case of VR experiences. Note that, while the terms “tutorial” and “training” are often used interchangeably, we will differentiate between them in our work. A tutorial refers to instructional material offered through step-by-step guidance. In contrast, training aims at gradually developing the skills necessary for a specific task in a game. Training is possible as a result of a series of tutorials, but tutorials do not necessarily cover the entire training procedure. Gameplay mechanics that require a certain level of training can be more intricate and are influenced by the user’s previous experience. In

 doi:10.20944/preprints202410.2102.v1

this study, we focus on one-time tutorials and do not include any mechanism for (potentially repetitive) training.

Tutorials are typically categorized into four types: non-interactive in-game, interactive in-game, background in-game, and no in-game tutorials [22]. Among these, in-game tutorials that integrate instructions throughout gameplay can have a significant impact on player learning [18]. Implicit tutorials are particularly effective in enhancing player understanding while minimizing boredom for experienced players, though the effects of game complexity and player type require further investigation [21]. Research indicates that tutorials significantly improve player engagement and retention, with their effectiveness influenced by game type, complexity, and player characteristics [15, 16,23]. Furthermore, the design of visual cues is critical in tutorial development, affecting both visibility and immersion [24,25]. In virtual reality (VR), tutorial design must account for sensory presence and text readability to maintain effectiveness [26–30]. While principles from traditional game tutorials are applicable to VR, additional considerations are necessary to prevent disruption of the user experience. Although manual tutorial design affords developers optimal control, it is labor-intensive and challenging to replicate efficiently. Lecuyer et al. [31] proposed an approach to simplify the generation of action sequences in virtual reality by recording each interaction's outcome and using this information to create a how-to-do tutorial. Knowing the game mechanics extracted is crucial for the tutorial generation methods [10,11]. To archive this, information from playtraces (event sequences recorded from playing the game) as well as information extracted from the code can be used [12]. Playtraces describe the actions that a player takes to complete the game. After both humans and AI agents had completed a series of games, the game progressions were compared, and those with the least number of unique mechanics were recruited for the next step. In the next step, a given game track was searched for the first occurrence of each game mechanic that appeared in the game track. These first occurrences were marked as critical mechanisms found and used to extend a Monte Carlo tree search agent. Results showed the playtrace information was useful in the sense that it matched human expectations of critical mechanisms better than the previously used method.

Our toolkit uses an approach similar to that of Lecuyer et al. [31] in conjunction with Green's method [12], where a playtrace can provide information on the critical mechanisms of a game. However, our approach does not require potentially problematic access to the game code. Instead, it gathers the necessary information implicitly by observing a human player. Critical actions are detected in a playtrace and speculatively checked for their contribution towards the game goal. With this approach, we will be able to provide a simple to implement tutorial generation without the limitation to precise interactions, as required by Lecuyer et al. [31]. To assess the functionality of our toolkit, we conducted an expert survey and evaluated our tutorial generator by implementing it in a puzzle game.

3. Assisted Tutorial Generator

The *assisted tutorial generator for virtual reality* (ATGVR) is a toolkit that can be applied to an arbitrary VR game to generate an onboarding level. The toolkit leverages playtraces, giving the designer the power to capture one or more success paths, convert them into tutorials, and present these tutorials to players. Developers may handpick particular playtraces and present the extracted tutorials in different game stages to showcase certain game mechanics; see Table 1. After setting up the toolkit, tutorial generation commences in four steps:

Step 1: Record a playtrace of the targeted game sequence from recording an experienced person playing the game.

Step 2: Automated analysis of events essential for the tutorial.

Step 3: Generate a tutorial by synthesizing visual cues for each of the essential events.

Step 4: Play back the tutorial to the end user.

The first three steps are explained in more detail in the following.

 doi:10.20944/preprints202410.2102.v1

Table 1. Different stages in the workflow of the ATGVR toolkit

	| | Initial Game Project Without Tutorial | |
I	**Initial Setup**
	Apply Framework Components Add tutorial Config Tool Add Tutorial Generation Event Manager Add Ghost (Agent)
II	**Repeated On Major Scene Changes**
	Developer Plays Game: Agent Plays The Game (once per event)
	Relevancy Detection Mode: Framework Observes Player Framework Records Events
	| | Tutorial Included in The Project | |

1- Recording a playtrace: Our system makes use of observing the player and recording events. We consider a potentially relevant action for the tutorial or moment that occurs in the game as an event. For the player's movement, we simply record an animation of the player avatar at run-time. In addition to motion, we consider the following event types:
- Collisions and triggers of all game objects (or a subset defined by the developer). As an example, the buttons in the play area can be observed by recording collisions.
- Controller inputs (SteamVR input events), such as grabbing and releasing game objects with the hands.
- A custom "game end" event. This event is special, as it only contains information about the terminal state of a game. It has to be raised manually in the game code when the game terminates and states what the final game state was (success or defeat).

2- Analysis of the playtrace: In this step, we analyze the playtrace to determine the information relevant to the tutorial. We use the extracted events and recorded player animations to let an agent play through the original game. Injecting events into the original game avoids the need for analyzing the game code, since we simulate game progress using the original game as a sandbox. The full playtrace is not suitable for generating a tutorial, as it typically contains a large number of irrelevant events. Since we are going to prompt the user of the tutorial to explicitly perform the relevant actions, the tutorial should consist only of the absolute minimum set of events required to achieve the goal. Hence, we analyze the playtrace to identify these essential events, while suppressing all others. This idea is conceptually similar to hyperstate space graphs [32], which collapse reversible actions into a single state. The goal of the analysis is thus to determine which events are irrelevant and can be omitted from the tutorial (see Figure 2).

Our algorithm iterates over the recorded events and greedily tests if the goal can be reached while the chosen event is suppressed. If the agent succeeds in reaching the goal even though the event has been suppressed, that event is removed from all future iterations. Otherwise, if the agent fails to achieve the goal, the event is retained. This process continues until only essential events remain. While the basic idea of suppressing irrelevant events is straightforward, the implementation of the suppression mechanism must ensure the causality of the game state is preserved. Traditional input events, such as pressing a button on the controller, can simply be discarded. However, events that affect other game systems, such as the physics simulation, may require custom event filters. For example, when an object is placed on another object, filtering out a single resulting collision event is not sufficient. In this case, the collision event would just be delayed to the next frame. Instead, the observed instance of a collision event has to be filtered for the entire analysis.

 doi:10.20944/preprints202410.2102.v1

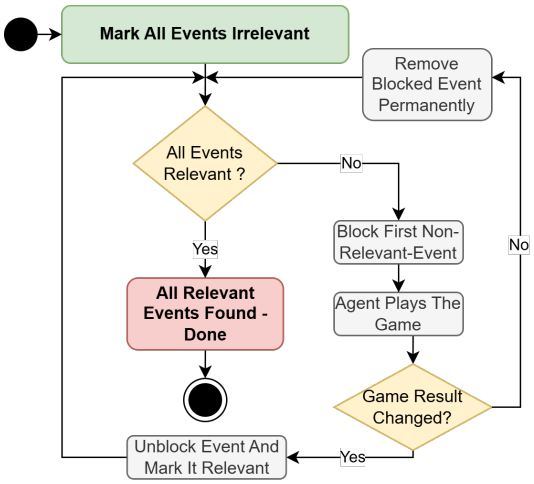

Figure 2. Algorithm for detecting event relevance

3- Tutorial synthesis: We synthesize a tutorial presentation from the events determined as essential in the previous step. For each such event, we first collect the participating primary objects, as well as the objects held in the hand at the time the event occurred. In addition, any secondary objects that collide with the primary objects are also collected. Secondary objects are important in showing the causes of the physical behaviors observed on the primary objects. For each object that participates in a tutorial event, a dedicated instruction step is synthesized. It typically consists of a looping animation of the involved objects and a textual description. The user must complete the step prompted in the instruction to advance to the next step of the tutorial. The condition to move to the next tutorial step depends on the type of event displayed, i.e., collision events require that a certain collision must occur; button events require a button to be pressed, and so on.

During playback of the tutorial steps, the *agent* is shown as a faded copy of a previous player, and faded copies of the objects involved in the instruction are shown. The faded copies are not participating in the overall game logic or physics simulation. Instead, all colliders and interaction components are removed; game objects only follow their predetermined animations. The animation loops replay what happened to them during the developer's playthrough. Only a short interval of the recorded animation is played, which revolves around the occurrence of a particular event. Typically, animation clips are around 5 s long and revolve around the event location. A clip plays 1 s before the event, pauses 3 s when the event occurs, and plays 1 s after the event. The pause serves to give the players time to read the displayed text and to show them exactly when the animation will be executed. As an example, when an object needs to be grabbed by the user, the tutorial shows the following steps: 1 s of reaching the object; 3 s of the closed grabbing hand on the object; and 1 s of dragging the attached object. The instructions are complemented by automatically generated text informing the user of what is expected at that point. The text expresses simple commands, such as "[object1] collides with [object2]", "press [button]", or "take [object1]". In the case of controller inputs, the integrated functionality of SteamVR is used to indicate which button to use, as most users will be familiar with the visuals built into SteamVR. Since auto-generated text often suffers from poor readability, there is an alias system that allows objects to be referred by an easy-to-understand name rather than the one used by the game engine internally.

4- Tutorial playback: The *tutorial runtime* is responsible for presenting the tutorial to the end user. It plays and stops the individual steps of the tutorial. In each step, it focuses on a single relevant event. For each such event, the tutorial runtime displays the prepared cues (animation, text) and waits for the expected action to be executed by the player. It continuously checks whether the conditions associated with the current event are met. These conditions depend on the event type. If the action is a simple button press, the condition to proceed is that the button is pressed by the user. For collisions and triggers, the corresponding event of the same colliders must occur. For picking up an object, the tutorial player waits for the object to be attached to the user's hand. When dropping an object, the

 doi:10.20944/preprints202410.2102.v1

final target position is matched. Additionally, to help the player find the drop-off location, a dashed line appears between the drop-off location and the current position, and a transparent copy of the object to the drop-off indicates the orientation.

Implementation: The current version of the toolkit is developed for the Unity game engine and the SteamVR API. In order to prepare a Unity project to use ATGVR, the developer needs to add the following software components to the scene: a configuration manager, an event manager, an agent, a tutorial runtime, and a modified SteamVR runtime. The *configuration manager* takes care of which components must be active in which step, as summarized in Table 2. These configuration changes affect the way the built-in animation recorder and collision detection work.

Table 2. ATGVR software components configuration in the four phases.

Component	Step 1	Step 2	Step 3	Step 4
Frequency	once	per event	once	as needed
Event manager	record	replay	replay	off
Animation recorder	record	replay	record	replay
Collision detection	on	off	off	on
Agent	off	on	on	off
Tutorial runtime	off	off	off	on
Modified SteamVR	on	on	on	off

The *event manager* handles all events that occur during the developer's playthrough, stores them in an SQLite database, and manages the iterations of the playtrace analysis. Events are all the collisions, triggers, and SteamVR input events that occur during the playthrough. Replaying events in this way relies on a *modified SteamVR* runtime, which accepts events to be submitted procedurally. Without code analysis, we do not know which functions are assigned to a particular event. As a workaround, we synthetically inject the original events to provoke the desired behavior. The modified SteamVR runtime also has an *animation recorder* and *collision detection* script attached to it. In this way, we can record the player's motion and track any collisions directly caused by the player. The player's controller inputs are observed by the event manager, which records actions reported by the built-in event system of SteamVR for later use. Thus, the agent is not only following a recorded trajectory, like in a conventional, non-interactive "ghost replay". Letting the agent interact with a game using the original input data is like replaying it. The agent triggers collisions and performs actions as the human user did in the recording. During playtrace analysis, a modified version of the agent is used. In this version, the recorded animations remain the same, whereas input events and collisions are disabled to determine what would happen without these interactions.

A game developer generates a tutorial by simply playing the game one time while going through the respective step via a dropdown menu. The menu also provides the option to disable the tutorial generator entirely, allowing for debugging without unloading ATGVR. Furthermore, developers can adjust the offset time to control when pausing occurs within an event. This offset ensures that animations are appropriately paused when an object is grabbed, preventing situations where the object has not yet snapped to the player's hand. To streamline the tutorial experience, a checkbox is available to collapse multiple collisions or trigger events between the same objects into a single tutorial step. This feature is important since collision and trigger events are evaluated for the entire playthrough, rather than on a per-occurrence basis.

4. User Study

To evaluate the performance and usability of the ATGVR toolkit, we conducted a between-subjects user study in a VR puzzle game. Conducting a between-subjects user study allows us to evaluate the

 doi:10.20944/preprints202410.2102.v1

user experience based on a single level of the tutorial. Participants are assigned to encounter tutorials generated by our toolkit or manually developed tutorials. This approach prevents the accumulation of the effects of multiple tutorial levels on user learning, allowing us to assess the impact of each tutorial generation method independently. Since the current version of the toolkit is primarily designed for physics-based interactions and locomotion, our decision was to create a physics-based puzzle game. Our evaluation focused primarily on the cognitive skills of the participants, rather than the abilities obtained with physical training. In addition, we intentionally favored short gameplay durations to minimize the impact of physical demands on user performance. Given the toolkit's current focus on individual users, we conceived the idea of an escape room game. In this game, we developed a dominoes-like puzzle to escape the room, with a sole win condition. This game scenario was chosen to diminish the influence of training on user experience and place greater emphasis on the tutorial aspect Since the main focus of the ATGVR toolkit in this study is learning the gameplay in VR games, we considered an introductory level for new users to help them learn the basic controller interaction and locomotion system. Consequently, our study consisted of four main levels: (1) general introduction level, (2) onboarding level, (3) challenge level, and (4) control level. At the general introduction level, participants learn about the VR controller and basic interactions. We designed this level to reduce the gap between experts and new users and to reduce the bias of the study results due to prior experience. The onboarding level is the main part of the tutorial in this study and has two variants, *Assisted* (the tutorial steps are generated using ATGVR) and *Manual* (the tutorial steps are created manually by the developers). Participants were exposed to one of these variants. To avoid bias in the design of the two variants, we let two different individuals create the onboarding levels. One of the authors created *Assisted* purely by applying the ATGVR toolkit, while *Manual* was created by a different person without any help from the ATGVR toolkit. Both creators did not communicate their design ideas before completing their respective onboarding levels. The challenge level is a puzzle game that measures the user's performance after the onboarding level. The control level is similar to the onboarding level, except that the user experiences the variant of the tutorial that they did not already experience in the onboarding level (i.e., *Assisted* players see the *Manual* level and vice versa).

We attempt to answer the research questions raised in Section 1 by defining in-game measurements, pre-questionnaires, and post-questionnaires. Our goal was to find out how the onboarding experience provided by *Assisted* compares to that of *Manual* with respect to user performance and preferences.

Materials: Before the study, we asked users to complete a demographic questionnaire about their age, gender, experience with games/puzzles, and VR. During the study, two types of measurements (objective and subjective) were collected. As objective measures, we recorded the duration of each level (onboarding, challenge, and control levels). In addition, we counted the number of times users invoked the reset button (which returns the game to its initial state to try a new solution). As subjective measures, we asked users to complete the System Usability Scale (SUS) questionnaire, a 10-item questionnaire measuring the learnability and usability of the application, the Igroup Presence Questionnaire (IPQ), a 14-item questionnaire measuring sense of presence throughout the experience, and the NASA TLX questionnaire, a 7-item questionnaire measuring task load. Furthermore, after the entire experience, we asked participants to complete a questionnaire about their personal preferences and any recommendations for further improvements.

Setup: Since the current version of the toolkit is developed for the SteamVR plugin in the Unity game engine, we designed a VR experience in Unity 2022.1. In this game engine, we had the option to choose between different rendering pipelines: Standard, Universal, and HD. To make the project compatible with a broader range of devices, we chose the universal rendering pipeline. The VR experience was played on a PC with 32 GB RAM, AMD 5800X CPU and NVIDIA RTX 3080 Ti GPU, running Microsoft Windows 11. We used Oculus Quest 2 as HMD, since it provides good resolution per eye (1920×1832 pixels), hand tracking, and wireless and wired connectivity. To improve the user experience, we

 doi:10.20944/preprints202410.2102.v1

connected the Oculus device to the PC using AirLink over WiFi 6. In this way, we avoided having users deal with cables.

Participants: We conducted this study with a group of 30 participants (4 female, 26 male) in two locations, the university campus and an office outside the university campus. To make a comparison between subjects, participants were randomly assigned to two distinct groups. The age of the participants ranged from 23 to 65 years, with an average age of 30.5 years. Most of the participants (29/30) had a college degree or were studying at a university. Most of them (22/30) worked/studied in the field of computer science. Approximately half of the participants (16/30) had corrected vision (glasses), and one participant reported color blindness. In the demographic questionnaire, we asked users to tell us about their prior experience in games/puzzles and VR.

Procedure: At the beginning of the study, the user was asked to complete a demographic questionnaire. After completing this preliminary questionnaire, a dialog screen appeared asking the participant to put on the VR headset. Since we measured the duration of each level, this step helps us to ensure that users had enough time to put on the headset and be ready to use the VR application. At this stage, the general introduction level was loaded. The instructions in this level help the user learn the interactions of grabbing, throwing and pressing buttons (all interactions required for the gameplay). After successful level completion, the onboarding level was loaded. The users were divided into two groups and randomly assigned to one of the two variants of the onboarding level (*Assisted* condition and *Manual* condition). The game is a physics puzzle (similar to Dominoes) where the user is supposed to place objects in the appropriate slots. The tutorial helps participants by showing them the current object, the type of interaction, and the position of the slot. Since *Assisted* and *Manual* were designed by different developers, the visual representation of the hints and the elements of the tutorial noticeably varied; see Figure 3. In *Assisted* , there is a panel on the wall that displays the current task with text. This text can be helpful if the user is confused and cannot find the next object position. In *Manual* , the wall panel shows a simplified game goal (shows the initial state and the final state), see Figure 4. The onboarding level ends when the puzzle is solved, and the user immediately enters the challenge level. The challenge level is a puzzle game that has elements similar to the onboarding level that helps users focus on the gameplay rather than examining the elements and interactions. However, we increased the difficulty of the dominoes game at the challenge level and made it more challenging (using a folding path with two platforms and more obstacles). The final goal in the challenge level is to release a blue balloon in the room that stands out in color from the rest of the scene to avoid any ambiguity. If participants were able to solve the puzzle in less than 15 minutes, the duration of the puzzle solution was recorded, and they were considered successful.

Figure 3. A comparison between the visualization of tutorial hints in *Assisted* (left) and *Manual* (right) levels

In both cases, with successful and unsuccessful completion, a message was displayed in VR asking participants to remove the headset and complete the post-questionnaires (NASA TLX, IPQ, and SUS). After completing the questionnaire, we asked participants to put on the VR headset and play the control level. The idea of this level is to present the alternative tutorial design to the participants and ask them to state their preferences. In the user preferences questionnaire, we asked participants

 doi:10.20944/preprints202410.2102.v1

to give their opinion on the practicality of the tutorials, their choice about the design aspect, and recommendations for future developments.

Visual Cues: In this study, we used different kinds of visual cues [25] to present tutorial elements in the game environment. Among the main elements of both tutorial designs are integrated cues. They serve "go", "look", or "discover" purposes [25], see Figures 3, and 4. An example of these integrated cues for the look task is a circular arrow projected from the camera location on the ground; see Figure 4 right. This implementation can help the user find the next task, even if they get confused in a complex virtual environment. In addition, we used different kinds of emphasized visualizations as markedness factors, for example, to highlight interaction-enabled objects and placeholder location (see Figure 3, left). We also included some subtle cues in the game scene, such as the blue balloon (as the final goal), which stands out in color from the rest of the scene. However, in this study, we did not include any overlaid cues to avoid any possible spatial disorientation due to the fixed position of the user interface in the VR environment.

Figure 4. Examples of integrated visual cues: (left) The back wall shows hint board in *Manual* level, (right) circular arrow in *Assisted* level shows the look purpose

5. Results

Results from the user study: To assess the effectiveness of ATGVR, we analyzed user performance in the "challenge" level. We measured performance using two factors: the success rate and the duration of the level. The success rate is a binary metric that indicates the impact of the tutorial level on learning the game mechanics. The duration of the level is a continuous value that reflects the level of proficiency in solving puzzles. For both measures, we expected that *Assisted* is equivalent or better than *Manual* , but not worse.

A comparison between the success rate of *Manual* (81%, 13/16) and *Assisted* (93%, 13/14) shows that the average success rate is higher for the *Assisted* group. However, The Z-Test for two population proportions shows no significant differences between the two groups ($z = 0.933$, $p = 0.3524 > 0.05$). Another important measurement is the duration of solving the puzzle in each condition. We can assume that this value represents the influence of the onboarding level on the user's learning of the game mechanics. Figure 5 (Top) shows the violin graph of the duration of solving the puzzle for each condition in the challenge level. The mean duration for *Assisted* is 5.75 minutes (SD = 2.92, median = 4.98) and, for *Manual*, 6.18 minutes (SD = 3.24, median = 4.54).

To answer the first research question, we need to figure out whether *Assisted* needs significantly more (or less) time to solve the challenge level compared to the *Manual*. We defined the following null hypothesis: There is no difference between the performance of *Assisted* and *Manual*. Accordingly, the alternative hypothesis is as follows: The performance of the *Assisted* group is not equal (i.e., better or worse) to *Manual*. We used the Shapiro-Wilk test to analyze the variance for normality. The results showed that the duration of the level is normally distributed for *Assisted* ($p = 0.245$), but not for *Manual* ($p = 0.025$). Therefore, we used the Wilcoxon signed rank test to compare two samples. The results indicate that there are no significant differences ($p = 0.57$) that show that the *Assisted* group needs more time than the *Manual* group. Furthermore, we did not find any significant results ($p = 0.49$) showing that the *Assisted* group requires less time than the *Manual* group. Taking into account these results, the null hypothesis cannot be rejected. Unfortunately, there are still two possible interpretations: Either both groups are statistically equal, or the sample size is too small

 doi:10.20944/preprints202410.2102.v1

to reveal significant differences. To investigate the desired first interpretation, we need to find a statistically significant equality (non-inferiority) of *Assisted* and *Manual*.

To this aim, we defined the following null hypothesis: The performance of *Manual* is better than that of *Assisted* , i.e., *Manual* prepares a user to solve the challenge in significantly less time than *Assisted*. The corresponding alternative hypothesis is as follows: The performance of *Assisted* is equivalent (or even better) than that of *Manual* , i.e., the difference between the duration of the puzzle solving in the two conditions is not too large. Note that a null hypothesis involving a strict inequality ("better" instead of "better or equal") requires a suitable statistical test. Hence, we used *two one-sided t-tests* (TOST) which specifically addresses equivalence/non-inferiority [33]. In TOST, an equivalence boundary (the minimum difference between the values defined as equivalent outcomes) is introduced, and two one-sided t-tests are used to reject the null hypothesis. If both tests are rejected, we can conclude that the null hypothesis is rejected, and the observed difference falls within the equivalence range.

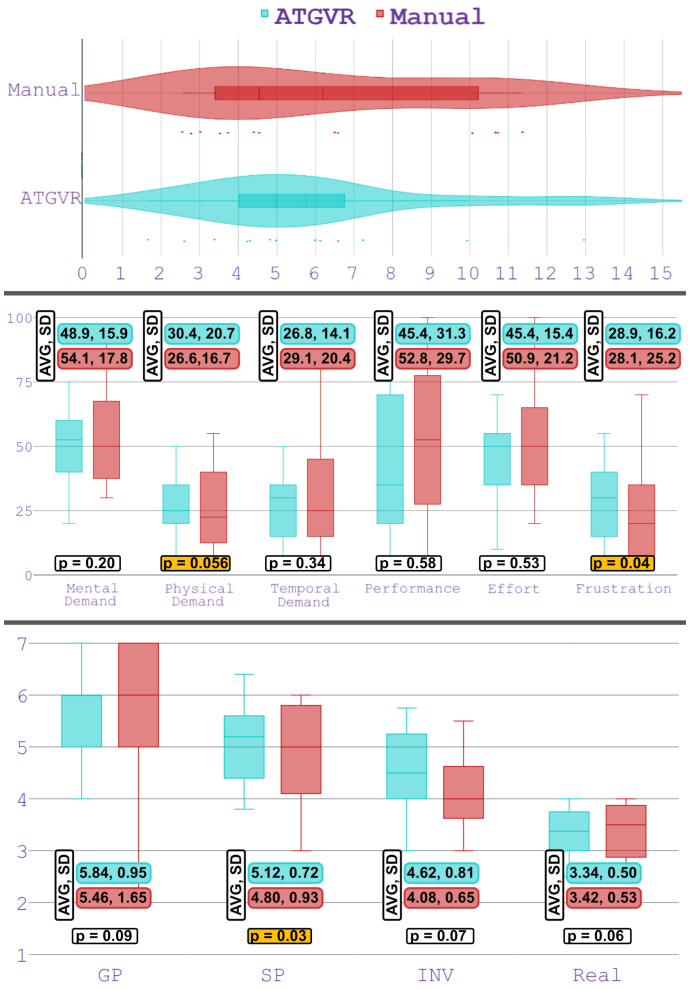

Figure 5. (Top) Violin plot of duration of solving the challenge level, Box plot of (Middle) NASA TLX and (Bottom) IPQ.

TOST assumes that the data are normally distributed. Therefore, we considered a hyperbolic tangent transformation to obtain a normal distribution. When performing TOST on the transformed data using an equivalence interval of three minutes, we found a significant result ($p = 0.043, t = 1.791$), which supports our expectation that *Assisted* and *Manual* are equivalent. Furthermore, when comparing the duration of the training level and the challenge level, there is a higher reduction in the duration of the challenge level for *Assisted* (mean reduction for *Assisted* : 54 s and *Manual* : 19 s).

 doi:10.20944/preprints202410.2102.v1

The results of the NASA TLX questionnaire are presented in Figure 5 (Middle). Given the nature of the game in our study, a VR puzzle game, it can be expected that mental demand, performance, and effort are generally high. Although the mean value of mental demand is lower for *Assisted* than for *Manual* , we found no significant differences between them. We found no significant difference in the "performance" item between *Assisted* and *Manual* , but, a significant non-inferiority in the case of "physical demand" and "frustration".

Figure 5 (Bottom) shows a comparison of the IPQ items between *Assisted* and *Manual*. Statistical analysis shows no significant difference in the cases of general presence (GP), involvement (INV), and realism (REAL) items of IPQ. The equivalence/non-inferiority test shows significant results for spatial presence (SP) with $p = 0.03$. The Shapiro-Wilk test for the SUS result shows that the data of *Assisted* are normally distributed ($p = 0.089$), but those of *Manual* are not ($p = 0.001$). In this case, applying the Wilcoxon test shows no statistically significant difference between the conditions ($p = 0.45$). From the user preference questionnaire, it appears that most participants prefer the *Manual* tutorial and find it more convenient (53%, 16/30). About 30% of the participants indicated that both tutorials were equally practical, and only about 17% found the *Assisted* tutorial more practical. However, looking at *Manual* separately, 31% chose *Assisted* and the same number chose *Manual* , while the rest rated both as equally practical.

At the end of the experience, we asked participants to complete a custom questionnaire containing two items. The custom questionnaire indicates that players who initially experienced the manual tutorial did not express a clear preference for either tutorial type. However, a notable trend emerged among the *Assisted* group, with a majority of the players indicating a preference for the manual tutorial. Specifically, seven players mentioned facing challenges in locating the next step within the auto-tutoring.

Interviews with developers: The viability of the tutorial generation process relies on both the effectiveness of the generated tutorials and the usability of the ATGVR toolkit itself from the developers' point of view. To gather insights into the developers' perspectives, we conducted interviews with three game developers. All three developers rated ATGVR tutorial generation as significantly less time-consuming than manual tutorial creation. They also highlighted the ease of integrating the toolkit into a regular game development workflow. Additionally, they expressed that the toolkit's usage was straightforward, and they recognized its potential, particularly for small indie game studies or early-access games that may have limited resources available for manual tutorial design during the initial stages of development.

Two of the developers expressed a high likelihood of using a system like this. The third developer, who was undecided about the likelihood of use, emphasized the importance of having more control over the resulting tutorial. They specifically mentioned the desire to use the generated tutorial as a foundation, as starting tutorials from scratch can often be a tedious process. They expressed the need to augment the generated tutorial with custom visuals and manually adjust the steps according to their specific requirements. Two of the developers rated the ATGVR tutorial generation as easier than traditional manual tutorial creation, while one developer remained undecided. In general, all developers recognized the potential of the ATGVR toolkit and considered it useful. However, given the limited sample size of the developers interviewed, future research should conduct a more comprehensive study to explore how developers perceive the ATGVR process and identify further areas of improvement from a developer's standpoint.

6. Discussion

The analysis of the measurement results with TOST shows the significant equivalence of solving puzzles at the challenge level for an equivalence interval of 2.5 minutes. This result reveals that if we consider the 2.5 min difference in the challenge level completion as equivalent (at about 6 min total duration), the assisted tutorial performs equivalently to the manually created one. However, equivalence/non-inferiority cannot be shown for an equivalence interval of less than 2.5 min. Another

 doi:10.20944/preprints202410.2102.v1

observation in favor of *Assisted* is that its success rate is slightly higher than that of *Manual*. In other respects, we can assume that *Assisted* works as successfully as the *Manual* tutorial.

Since the main activity in this VR experience is solving the puzzle, the NASA TLX questionnaire is expected to have high scores for mental demand, performance, and effort. However, the mean values of these items do not exceed the midpoint of the NASA TLX range of values. Even in a demanding task such as solving our VR puzzle, it seems that the tutorial helps the player achieve reasonable results in terms of performance without requiring great effort and mental challenges. Physical demands are relatively low, because participants did not require extra effort to solve the task and did not require unnatural locomotion, such as teleportation (they can move around the entire scene with just physical movements). Regarding temporal demand, users reported that they did not feel time pressure and did not require too much rush. Note that we did not inform users that there was a time limit for solving the puzzle, and we did not include a visible timer/clock in the scene. This design decision helped users play the game without feeling time pressure.

The IPQ results show a relatively high value for general presence. This item of the presence questionnaire is a generalized question about the sense of being there. One of the interesting results in IPQ is that the involvement item for *Manual* is medium; however, *Assisted* reports a significantly higher value. The difference could be due to the variations in the visualization used in the onboarding levels. As we used a stylized visualization in our VR environment, it was predictable that the realism item of IPQ shows a relatively low value.

Finally, SUS shows a very high value that can be interpreted as ease of use and the learnability of both tutorials. Although we did not find a significant difference between the conditions in the case of SUS, the results of the user preference questionnaire show that most of the participants preferred the visualization of the *Manual* tutorial. Specifically, several users indicated that they preferred the graphical representation, text, and icons in the manually created onboarding room over *Assisted*. For example, in the *Manual* version of the onboarding space, we represent the object placeholder as a wireframe object in blue (a color that is different from the rest of the scene), while, in *Assisted* , we use a transparent placeholder. Most participants find the wireframe version to be more convenient and easier to see. Since it is possible to use most of the visualization elements of the *Manual* level in *Assisted* , we plan to improve *Assisted* according to users' recommendations for future work. Based on these observations, we answer our research questions:

RQ1: Regarding performance (required duration to solve the puzzle), the semi-automatically generated tutorial is equal (not inferior) to the manually created tutorial.

RQ2: In *Assisted* , 93% of the participants, and, in *Manual* , 81% of participants successfully solved the puzzle in the challenge level. However, the results of the statistical analysis indicate that there is no significant difference between the two groups.

RQ3: SUS and NASA TLX did not reveal any significant differences between conditions. However, in the involvement item of IPQ, *Assisted* reported a significantly higher value than *Manual*.

These findings suggest a strong support for *Assisted*. We may assume that semi-automatically generated tutorials (for example, for building a large collection of on-the-job training scenarios) are feasible and ready to be used in practical applications.

Limitation and future work: According to the insights gathered from the custom questionnaire, it is evident that certain visual cue choices may enhance the usability of the tutorial. As an example, it is believed that the bright blue colors used in the manual tutorial played a significant role in players' preferences, as they facilitated easier identification and enhanced visual clarity of tutorial steps, even from a distance. To address this preference gap in future iterations, alternative visual cues could be explored within the ATGVR tutorial. By implementing such visual cues, it may help bridge the preference gap by providing clearer guidance and improving the ease of progression within the tutorial. A natural limitation of our tutorial-by-demonstration approach is that the game logic must be deterministic. Consequently, only one chain of actions can be considered at a time. If multiple alternative ways of reaching a goal are presented, they will require multiple separate

 doi:10.20944/preprints202410.2102.v1

demonstrations. Likewise, random game behavior cannot be properly supported. Furthermore, feedback from developers suggests that incorporating the capability of developers to refine and fine-tune tutorials manually would greatly enhance the usability of this toolkit. These potential improvements can be valuable in improving the overall user experience and addressing the challenges mentioned by players. More experimentation with the visual design and cues of the ATGVR tutorial is warranted to refine the tutorial experience and narrow the preference gap between the *Manual* and *Assisted* tutorials.

7. Conclusions

In this study, we present a toolkit for assisted tutorial generation for VR games. ATGVR aims to simplify the process of creating such tutorials. Our toolkit can create gameplay steps according to the successful gameplay of the developer. It records required gameplay steps and extracts events required for successful completion while filtering out those that do not influence the gameplay.

We conducted a user study with 30 participants to evaluate the performance of the ATGVR toolkit. The results of this evaluation show that an assisted tutorial performs equivalently to a manually created one. Analysis of the results of the presence questionnaire shows that *Assisted* received a significantly higher score for the involvement factor. In addition, the success rate of the participants in *Assisted* was higher than in *Manual*. Hence, the ATGVR toolkit can be used as part of game development in order to reduce the time and budget needed to develop a tutorial for a game.

In this evaluation, we were limited by the number of participants. By expanding the study to more participants, we can expect a better understanding of the benefits and limitations of the ATGVR toolkit. Furthermore, considering the user experience in future studies, we can glean valuable insights into the acceptance and reception of each tutorial generation approach. The current version of this toolkit was developed for the SteamVR plugin in the Unity game engine, but we plan a wider coverage by extending it to OpenXR and the Unreal Engine as well. While the toolkit works on general game events, there may be limitations with respect to more complex game mechanics involving non-deterministic game event sequences or even game mechanics that do not produce any events. We plan to investigate these cases in future work.

References

1. Behmadi, S.; Asadi, F.; Okhovati, M.; Sarabi, R.E. Virtual reality-based medical education versus lecture-based method in teaching start triage lessons in emergency medical students: Virtual reality in medical education. *Journal of Advances in Medical Education & Professionalism* **2022**, *10*, 48.
2. Harrington, C.M.; Kavanagh, D.O.; Quinlan, J.F.; Ryan, D.; Dicker, P.; O'Keeffe, D.; Traynor, O.; Tierney, S. Development and evaluation of a trauma decision-making simulator in Oculus virtual reality. *The American Journal of Surgery* **2018**, *215*, 42–47.
3. Safikhani, S.; Pirker, J.; Wriessnegger, S.C. Virtual Reality Applications for the Treatment of Anxiety and Mental Disorders. 2021 7th International Conference of the Immersive Learning Research Network (iLRN). IEEE, 2021, pp. 1–8.
4. Chauvergne, E.; Hachet, M.; Prouzeau, A. User Onboarding in Virtual Reality: An Investigation of Current Practices. Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems; Association for Computing Machinery: New York, NY, USA, 2023; CHI '23. doi:10.1145/3544548.3581211.
5. Lin, H.C.S.; Yu, S.J.; Sun, J.C.Y.; Jong, M.S.Y. Engaging university students in a library guide through wearable spherical video-based virtual reality: Effects on situational interest and cognitive load. *Interactive Learning Environments* **2021**, *29*, 1272–1287.
6. Wu, H.K.; Lee, S.W.Y.; Chang, H.Y.; Liang, J.C. Current status, opportunities and challenges of augmented reality in education. *Computers & education* **2013**, *62*, 41–49.
7. Bradford, G.R. A relationship study of student satisfaction with learning online and cognitive load: Initial results. *The Internet and Higher Education* **2011**, *14*, 217–226. doi:https://doi.org/10.1016/j.iheduc.2011.05.001.

8. Miguel-Alonso, I.; Rodriguez-Garcia, B.; Checa, D.; De Paolis, L.T. Developing a Tutorial for Improving Usability and User Skills in an Immersive Virtual Reality Experience. International Conference on Extended Reality. Springer, 2022, pp. 63–78.
9. Putze, S.; Alexandrovsky, D.; Putze, F.; Höffner, S.; Smeddinck, J.D.; Malaka, R. Breaking the experience: Effects of questionnaires in vr user studies. Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, 2020, pp. 1–15.
10. Green, M.C.; Khalifa, A.; Barros, G.A.; Togellius, J. " Press Space to Fire": Automatic Video Game Tutorial Generation. Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference, 2017.
11. Cerny Green, M.; Khalifa, A.; Barros, G.A.; Machado, T.; Nealen, A.; Togelius, J. AtDELFI: automatically designing legible, full instructions for games **2018**.
12. Green, M.C.; Khalifa, A.; Barros, G.A.; Machado, T.; Togelius, J. Automatic Critical Mechanic Discovery Using Playtraces in Video Games. International Conference on the Foundations of Digital Games, 2020, pp. 1–9.
13. Green, M.C. AUTOMATIC VIDEO GAME TUTORIAL GENERATION. PhD thesis, NEW YORK UNIVERSITY, 2022.
14. Pfau, J.; Smeddinck, J.D.; Malaka, R. Automated Game Testing with ICARUS: Intelligent Completion of Adventure Riddles via Unsupervised Solving. Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play; Association for Computing Machinery: New York, NY, USA, 2017; CHI PLAY '17 Extended Abstracts, p. 153–164. doi:10.1145/3130859.3131439.
15. Andersen, E.; O'rourke, E.; Liu, Y.E.; Snider, R.; Lowdermilk, J.; Truong, D.; Cooper, S.; Popovic, Z. The impact of tutorials on games of varying complexity. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2012, pp. 59–68.
16. Morin, R.; Léger, P.M.; Senecal, S.; Bastarache-Roberge, M.C.; Lefèbrve, M.; Fredette, M. The Effect of Game Tutorial: A Comparison Between Casual and Hardcore Gamers; Association for Computing Machinery: New York, NY, USA, 2016. doi:10.1145/2968120.2987730.
17. Passalacqua, M.; Morin, R.; Sénécal, S.; Nacke, L.E.; Léger, P.M. Demystifying the First-Time Experience of Mobile Games: The Presence of a Tutorial Has a Positive Impact on Non-Expert Players' Flow and Continuous-Use Intentions. *Multimodal Technologies and Interaction* **2020**, *4*, 41.
18. Shah, V. Examining the Effect of Different Types of Tutorials on New Players of a Compute Science Teaching Game. PhD thesis, Northeastern University, 2018.
19. Vesa, M. The Importance of Fun in Video Game Tutorials: A Case Study of Uncharted 4. B.S. thesis, 2021.
20. Ramirez Sessarego, A.; Arévalo Arancibia, F. Making Sense of a Game: A look into Tutorials and Character Mechanics **2019**.
21. Cao, S.; Liu, F. Learning to play: understanding in-game tutorials with a pilot study on implicit tutorials. *Heliyon* **2022**, *8*, e11482.
22. Jamieson, D. 4 ways to teach your players how to Play your game, 2015.
23. Passalacqua, M.; Morin, R.; Sénécal, S.; Nacke, L.E.; Léger, P.M. Demystifying the First-Time Experience of Mobile Games: The Presence of a Tutorial Has a Positive Impact on Non-Expert Players' Flow and Continuous-Use Intentions. *Multimodal Technologies and Interaction* **2020**, *4*. doi:10.3390/mti4030041.
24. Bardzell, S. Systems of Signs and Affordances: Interaction Cues in 3D Games. *Extending Experiences. Structure, analysis and design of computer game player experience* **2008**, pp. 191–209.
25. Dillman, K.R.; Mok, T.T.H.; Tang, A.; Oehlberg, L.; Mitchell, A. A visual interaction cue framework from video game environments for augmented reality. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, 2018, pp. 1–12.
26. Lilius-Lundmark, R.; Torbjörnsson, D. Presence when interacting with Graphical User Interfaces in Virtual Reality-A Qualitative Study of Tutorials in Virtual Reality Games **2020**.
27. Rau, P.L.P.; Zheng, J.; Guo, Z.; Li, J. Speed reading on virtual reality and augmented reality. *Computers & Education* **2018**, *125*, 240–245.
28. Kojić, T.; Ali, D.; Greinacher, R.; Möller, S.; Voigt-Antons, J.N. User experience of reading in virtual reality—finding values for text distance, size and contrast. 2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX). IEEE, 2020, pp. 1–6.
29. Beier, S.; Berlow, S.; Boucaud, E.; Bylinskii, Z.; Cai, T.; Cohn, J.; Crowley, K.; Day, S.L.; Dingler, T.; Dobres, J.; others. Readability Research: An Interdisciplinary Approach. *arXiv preprint arXiv:2107.09615* **2021**.

 doi:10.20944/preprints202410.2102.v1

30. Fan, J.; others. Designing a tutorial for a multiplatform and multiplayer virtual reality game. PhD thesis, Massachusetts Institute of Technology, 2019.
31. Lécuyer, F.; Gouranton, V.; Reuzeau, A.; Gaugne, R.; Arnaldi, B. Create by doing–Action sequencing in VR. Advances in Computer Graphics: 36th Computer Graphics International Conference, CGI 2019, Calgary, AB, Canada, June 17–20, 2019, Proceedings 36. Springer, 2019, pp. 329–335.
32. Cook, M.; Raad, A. Hyperstate Space Graphs. *IEEE Transactions on Games* **2022**, *14*, 435–445. doi:10.1109/TG.2021.3095393.
33. Schuirmann, D.J. A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. *Journal of pharmacokinetics and biopharmaceutics* **1987**, *15*, 657–680.