Virtual Reality Course Project

Iñigo Gabirondo López (876333), Blanca Lasheras Hernández (736056)
Virtual Reality

May 22, 2023

1 Introduction

Within the last years, virtual reality (VR) has changed the way we experience interactive environments, with applications that span a great variety of fields, including the gaming industry. Our project focuses on the development of a VR archery application, aiming to apply the principles of perception, hardware, and content generation discussed in previous laboratory and theory sessions. Our purpose is to create an engaging and immersive experience by implementing a game set in the Wild Wild West. In addition to implementing the essential features, we have conducted user experiments to explore phenomena like change blindness and infer valuable insights of the user's behavior. In addition, we have tried to optimize the overall gaming experience by adapting the game and adding extra features.

In this report, we provide an overview of our path through developing the different features for our application: We discuss theoretical foundations, technical implementation details, and user experiments, offering insights into one of the state-of-the-art applications of VR. Also, we spotlight the potential applications of VR-based techniques and applications, as well as their ability to create captivating interactive experiences.

2 Archery application

2.1 Application overview

Through this work. we have implemented a first-person game that is inspired by spaghetti western movies. There, the user has the objective of shooting the target, which in this case is a bandit, with the use of a bow and some arrows that they can interact with. For that purpose, we have created three versions of the game: We have first developed a *demo* where all main functionalities were implemented and tested to further be included in the main application. After that, we created a *desktop* application that would serve as a preliminary prototype for testing the whole game experience and experiments. Finally, by changing the controls and canvas of the *desktop* application, we exported it to a *smartphone* application (see Fig. 4).

Regarding the final application (*smartphone*), it can be played in both English and Spanish, it allows two game modes (exploration and experiment) and when any of these two games finishes, it displays different game data of the player. The application starts by showing an initial menu that enables selecting the language and the game mode, after that it passes to the game scene, and when the game time finishes, it shows a game over scene with different data about the game.

2.1.1 Baseline [MT-0]

First of all, we implemented the base features of the archery application. An overview of these features is given in this section.

2.1.2 Interaction

For developing our app, we have developed a *desktop version* that allowed us to test all the functionalities, checking if they had been correctly implemented. Once our game worked correctly, we have adapted it to be played in our VR headset and controller, where the rotation of the user is controlled by the phone's gyroscope, and the joystick allows translation in local coordinates. In addition, buttons [1] are used for pointing towards the target and shooting, as well as for interacting with the bow and arrow (further details can be found in Section 2.2.1).

2.1.3 Arrow trajectory

For our game, we have designed the arrows to have a physically-based behavior. For that purpose, we have defined them as rigid bodies that follow the laws of physics. Therefore, when launching them at a specific force, they draw a parabolic path towards the target point.

2.1.4 Arrow collision

For achieving that arrows remain stuck on the point they hit, we have attached colliders to them, which allow defining the shape and physical properties of objects. They allow interactions (i.e., detecting collisions and responses) among game objects and the environment in a realistic manner. They can have different shapes, such as boxes, spheres, capsules, cylinders, etc. By attaching a collider component to a game object, you define its physical boundaries and enable it to participate in collision interactions. By attaching colliders to the different game objects implied (i.e., floor, target, arrows, objects, etc.), Unity's physics engine can detect a collision between them and provide collision data, such as the point of contact, normal direction, and other useful information. In our

[1] In our experience, we have seen that buttons differ according the phone device paired with the controller. More information can be found in Appendix B.

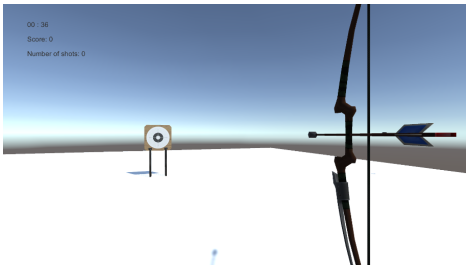

Figure 1: *Demo* version.

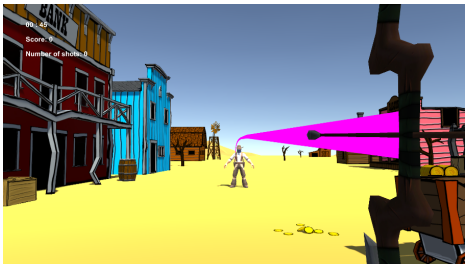

Figure 2: *Desktop* version.

Figure 3: *Android* version.

Figure 4: Implementation process of our game.

case, this allows us to implement some game mechanics, by determining whether an arrow hits a target or not.

In our case, we have implemented arrow collision, and the arrows stay in their position until the user gets them back. For our desktop version, we first implemented a logic where the user had to be close to the arrow they want to take, and then press a key. However, for more playability, we decided to limit it to distance and position with respect to the arrow. Therefore, in both our desktop and VR-headset versions, the user needs to get closer to the arrow and look at it in order to take it, and then, they will be able to shoot it again.

2.1.5 User movement and distances

We have implemented translation and rotation of the user in the *demo*, *desktop* and *HMD-based* version. In the computer versions, users could translate and rotate with WASD keys and mouse respectively, whereas in the headset version, the translation is achieved by the controller's joystick and the rotation is calculated by the mobile phone's gyroscope.

2.1.6 Feedback

We have also implemented a functionality so that, when the user succeeds or fails in hitting the target, feedback is provided. In our case, we have decided to implement auditory stimuli to increase the imnersion, as well as to augment the user's engagement by using humoristic sounds that make them want to hit the target again. We have associated the sounds to the collision results: This is, depending on the label of the game object the arrow collides with, a different message is emitted.

2.2 Addditional features [OT-0]

Once we have our core application complete, we have decided to implement additional features that improve the gaming experience.

2.2.1 Bow and arrow interaction

Similarly to the behavior explained in Section 2.1.4, we first implemented the functionality of taking and leaving the bow for both our desktop and HMD versions by pressing a key near them. However, as we explain in Chapter 3, we needed to add some modifications to our desktop game in order to perform experiments related to the assigned extension. Therefore, in our last version of the application, the only flag for taking the bow and arrows is the user's proximity to them.

This behavior is additionally modified for performing user experiments (see Section 3.1), where the bow is diectly attached to the user, and the arrows are charged automatically when the user turns to the *charging position* (i.e., within a range of longitudes).

2.2.2 Target movement

We have implemented some target movement. In our initial *desktop* version, to increase the difficulty of our game, the bandit starts moving when user presses key `m`. However, for the sake of simplicity, and to avoid overloading the user with additional commands, we have set the bandit to move periodically from side to side after he's shooted twice.

2.2.3 Game menu and languages

We believe that opening an aplication and directly starting playing can result shocking for some players. Therefore, we have implemented a very simple menu that allows switching languages and selecting between the exploration and experiment game modes. The different scenes of the game can be seen in Figure 8.

As far as the languages are concerned, most of the people that have tested the game are non-english speakers, so playing the game in english may suppose an extra difficulty for them. Hence, we decided to translate the full game to Spanish.

2.2.4 Scoring system

For our game, we first designed a ranking system where the player introduced an ID and could see their scores after playing, ordered in a ranking that took previous games in account. However, due to the parameters we wanted to measure as part of the study of our extension topic, we decided not to provide a reference score for players, since we thought that competing against known thresholds could influence user's behavior

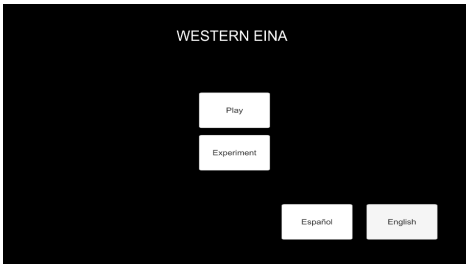

Figure 5: Initial menu.

Figure 6: Game mode.

Figure 7: Game over scene.

Figure 8: Different scenes of the game.

in a significant manner [2]. That is why we decided to implement a scoring system where the player can visualize their own statistics but not others'. That way, players do not have a reference and they just compete against themselves. We find this particularly significant since each user plays the game a number of times for our experiments (further details of the experiments procedure are discussed in Section 3.1.3).

2.2.5 Ghost trajectory

For drawing the trajectory of the arrow, we define `DrawTrajectory()` function, that renders a line with a specified number of points by calculating the position of the impact point through the rotation angle in the x axis. Then, the position of each of the specified points is set according to the velocity of the arrow, which depends on its mass. This functionality gets active when the user has an arrow and presses the button for shooting. The trajectory keeps plotted while the user holds the key, so they can adjust it to the target, and it disappears when the user releases it and therefore shoots the arrow. For performing this holding action without overloading the main `Update` thread, we set a coroutine where we perform a loop where we repeatedly call the `DrawTrajectory()` function in a different thread, thus avoiding computation overheads. Then, we start the coroutine when the user presses the button, it runs while the user holds it, and it stops when the user releases it. The generated ghost trajectory can be seen in Figure 2.

2.2.6 Auditory stimuli

As we have explained in Section 2.1.6, we provide auditory stimuli as feedback for the user depending on wether they hit the target or not. Additionally, we have included an audio source to the main camera (i.e., the user) that provides a greater ambience of the game, increasing the immersive experience for the player [9]. For this game, we have used royalty-free music and sound effects [11].

[2]Url to the Google Forms that gathers the data: `https://forms.gle/6WGGMiHZkjUXG9hA7`

3 Change Blindness [MT-1]

Change blindness is a phenomenon in which humans fail to notice changes in their visual environment, even when the changes are significant or occur right in front of them. This psychological occurrence happens because our attention is selective and limited: We tend to focus on specific aspects of our environment while ignoring others. When a change happens outside our focus of attention, we often fail to notice it, even if the change is substantial and obvious [12].

There are several works that address different aspects that influence this behavior, from a physiological [2] and neurological [4] point of view, studying the influence and limitations of the visual working memory in our brains' capacity to retain some information. Some works address these impacts by studying the influence of the scene through the background complexity [6], the number of manipulations [1], or the complexity of an object [14], among others. Martin et al. [8] propose a systematic study of the change blindness effect by simulating natural viewing conditions through VR environments in 3D, where they explore several of the aforementioned factors among others such as the *field of view* (FoV) or the distance of the manipulated object. However, change blindness effect is still not completely understood, therefore there are some limitations and open challenges that can be addressed. In our case, we have decided to focus on studying change blindness effect when the observer is under high cognitive load (i.e., they have been asked to perform a specific task).

Despite there are previous works that consider people carrying out tasks such as walking [13], these do not demand a great ammount of cognitive resources: They are performed by the brainstem, which is the most primitive part of the brain. Our work proposes a different approach where we study the impact of adding cognitive load to the observers. In addition, we analyze the influence of informing the users that changes are taking place in the scene.

3.1 Methodology

3.1.1 Manipulations

For our experiment, we have decided to apply four different manipulations [8] in the same outdoor synthetic environment ambienteted in a Western manner:

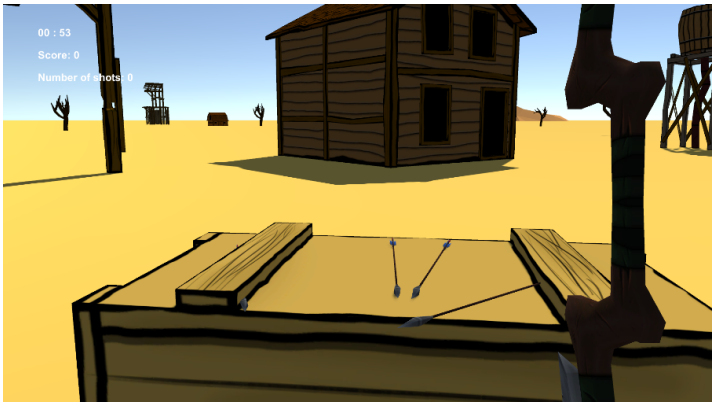

Figure 9: View of the charging zone where the user can recharge arrows.

1. *Addition* We insert an element in an empty location in the scene.
2. *Removal* We remove an element from the scene.
3. *Relocation* We move an existing object to a different location in the scene, or there is an exchange of positions between two elements of the scene.
4. *Replacement* A different element replaces an existing one in the same position.

We have implemented 16 changes (four of each kind) that apply in a randomized order for each user experiment and do not repeat for the same user. We apply one change at the same time (see Section 3.1.3), and all the changes happen in the shooting FoV. Manipulations occur when the user is not looking at the shooting FoV and is looking within a charging range of longitudes, as explained in Section 2.2.1. Further implementation details are provided in Section 3.1.3.

3.1.2 Participants

We have carried out the experiments with seven participants: 3 identified themselves as female, 4 identified themselves as male, and none of them identified themselves as non-binary, other, or preferred not to say: The average age is 35.5 years, with a $STD = 18.32$. All participants took the experiments voluntarily. One subject reported presbyopia, but further analysis showed no significant differences in behavior compared to participants without presbyopia. The remaining participants reported normal to corrected-to-normal vision. 100% of the subjects had never used an HMD before the experiment. 84.7% reported the highest score in self-perception of well-being, dizziness and general sensation in a five-score scale, whereas 14.3% reported the second highest score.

3.1.3 Procedure

We have carried out the experiment by sitting the users in a rotating stool to prevent sickness caused by unwanted translational movements. We have divided our experiment in four phases for each user. In addition, at the beginning of the experiment and after each phase, the user has to fill a brief survey with qualitative information we use for analyzing their behavior (see Chapter 4) and the studied effect. We also help the users put the HMD and explain them how to adjust it. Then, we provide them the controller and explain them how to use it to interact with the virtual environment.

First, the user conducts an **exploration** of the scene where they have 60 seconds to freely move within the scene with the controller and head orientation. The user is told to try a new game with the objective of getting familiar with the scene and exploring it. Also, an explanation of how to grab the bow and arrows, as well as shooting, is provided. Then we start performing the change blindness experiment itself. For this purpose, we implement some modifications to adapt the game to our experiment's requirements as explained below.

For the **first phase**, the user is told that they will be placed in a fixed position, only being able to rotate. Their objective is to achieve the highest possible score in 30 seconds. In order to do so, it is necessary to shoot, turn around to the charging area to automatically recharge another arrow, and turn back to the original position to shoot again. The user is not notified about the happening changes until the experiment ends, where they are asked to report any changed in the scene, if noticed. 100% of users reported not noticing any changes in the scene.

The **second phase** consists of performing the same task of achieving the highest result but informing the user that now they may encounter changes in the shooting area in 30 seconds. They are aware that only one change per shoot is performed, and that they occur when they are not looking at the shooting FoV (i.e., when they are charging an arrow). We encourage them to notify us if they find any of the produced changes, and we also gather their responses in the form after this phase ends.

Finally, for the **third phase**, we encourage the user to directly detect as much changes in the scene as possible during 30 seconds, no matter the score they obtain. However, since changes happen when they are charging arrows, they still need to shoot and turn. After carrying out the experiment, users fill up the form [3] with the detected changes as well as other questions related to our study.

Changes to [MT-0] For carrying out the different phases of the experiment, we have had to perform a number of modifications in our application. First, in order to perform changes for our experiment, we needed to control the FoV that the user sees. That is why we limited the user's movement to only rotation by turning arround themselves. In addition, to simplify the user's interaction with the environment and avoid biases due to aspects such as gaming experience or age, we automatically provide the bow since the beginning, and the way to recharge arrows is by turning to the charging zone (see Fig. 9). Therefore, for the experiment the user will not

[3] Url to the experiment survey: `https://forms.gle/3r6MLu1RYjwYWm3h6`

Figure 10: Detection ratio per each type of modification. As it can be seen, relocations are the most seen, whereas removals appear to be the less distinguished ones.

be able to gather the shooted arrows. In addition, regarding the target, it does not move until it gets hit twice. Then, it starts a periodic movement, as explained in Section 2.2.2.

3.2 Evaluation

Metrics For evaluating our results, we have evaluated the *detection ratio* as the correlation between the amount of times a manipulation has been correctly detected and the total times that manipulation has actually occured. Then, we also analyze the detectabillity of the four types of manipularion mentioned above.

Results In Table 1 we present the detaction ratio obtained for the different manipulations. It is important to notice that no detections have been made in the first phase of the experiment. We believe this is due to the lack of prior notice that changes were happening. This leads us to an interesting insight: Users may encounter more difficulties finding changes in a scene, even if they appear to be obvious, if they have not been previously notified. In addition, we can see that, in most cases, detection rates increase when the user is task-driven to directly find the differences, as requested in the third phase of the experiment. Therefore, guided by the obtained results, we believe the change blindness effect increases when the cognitive load increases, as it happens in the second phase, where users are encouraged to obtain the highest score they can achieve.

Moreover, we have analyzed the detection ratios by type of manipulation (see Fig. 10), where we can see that relocations are the most percieved effects. Specially, the ones that imply bigger objects such as buildings. This can be checked by looking at Table 1, where the most seen change appears to be change number 9, which implies the relocation of two buildings of different colors, followed by change number 11, which implies exchanging positions between a tree and a windmill.

Table 1: Detection ratio per each of the experimental phases, as well as for the total ammount of manipulations performed in the experiments.

Change	**Detectability** (↑)			
	P1	*P2*	*P3*	*Total*
0	0.0 %	50.0 %	0.0 %	16.67 %
1	0.0 %	0.0 %	0.0 %	0.0 %
2	0.0 %	0.0 %	25.0 %	16.67 %
3	0.0 %	0.0 %	0.0 %	0.0 %
4	0.0 %	0.0 %	0.0 %	0.0 %
5	0.0 %	0.0 %	0.0 %	0.0 %
6	0.0 %	0.0 %	0.0 %	0.0 %
7	0.0 %	33.33 %	20.0 %	18.18 %
8	0.0 %	0.0 %	0.0 %	0.0 %
9	0.0 %	**66.67** %	**100.0** %	**60.0** %
10	0.0 %	0.0 %	0.0 %	0.0 %
11	0.0 %	33.33 %	40.0 %	30.0 %
12	0.0 %	0.0 %	0.0 %	0.0 %
13	0.0 %	0.0 %	50.0 %	12.5 %
14	0.0 %	0.0 %	0.0 %	0.0 %
15	0.0 %	0.0 %	50.0 %	16.67 %

4 Behavior analysis [OT-1]

VR has emerged as a powerful tool for both understanding and analyzing human behavior in several application domains. VR enables studying behavior in controlled inmersive and interactive environments while measuring different parameters that can provide information aboyt different psychological principles proper from humans. In our case, we have decided to cover this topic as an additional extension to complete the information of the user studies we have been performing by providing additional insights of the impact of augmenting the cognitive load, also in the opposite direction from the previously studied one. Some works explore users' visual attention by analyzing and even generating scanpaths [10]. Other works focus on visual saliency as a key indicator of the user's visual attention [3, 7], also studied in task-driven environments [5]. Our work in this project has a specific aim to study how a higher cognitive load influences users' performance. Additionally, and due to the exceptionallity of our subjects, we have a 100% of users that have never used an HMD before. This allows us to gather other qualitative data on equal conditions in terms of experience.

4.1 Methodology

For our study, we have gathered differend kinds of information, in order to perform both qualitative and quantitative analysis. As explained in Section 2.2.4, we have stored the user's game statistic after each experiment to be able to measure their accuracy and performance. For that purpose, we have collected the number of shots they have carried out, as well as the number of succesful and failed ones. In addition, we have also asked about a personal evaluation of the diffi-

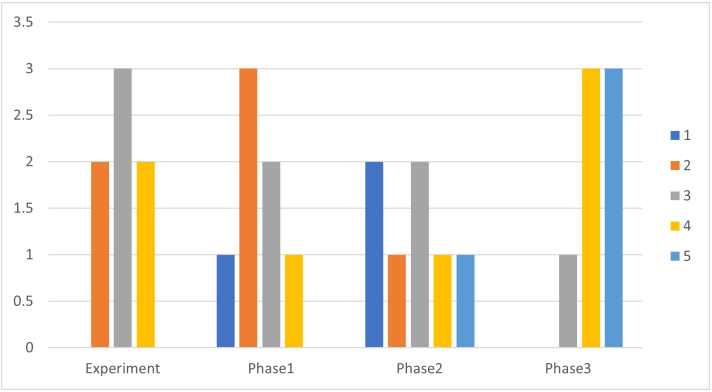

Figure 11: Level of difficulty reported by users. In scale from 1 (easy) to 5 (difficult).

Experiment	Mean accuracy
1	0.78
2	0.87
3	0.75

Table 2: Mean accuracy of users for experiments 1, 2 and 3.

culty of the game in a five-levels scale (results can be seen in Fig, 11).

4.2 Results

After performing an analysis of the obtained results, we have been able to see that the subjective level of difficulty increases as the time passes. We believe this is due to both the time passed and users' fatigue, which has also been stored, and the difficulty of the commanded task: In the first experiments, users did not know that changes were happening, whereas in the last one, they are focused in detecting changes that may be difficult to encounter (see Fig. 11).

In addition, we have evaluated the users' mean accuracy for each of the experimental phases. We have seen that the second phase is the one with higher accuracy. We believe this is due to the gain in experience, after getting used to the new environment, as well as to the less importance of the detecting-changes-related task (see Table 2).

Finally, it has been also interesting that users have used different strategies in order to try to find the changes of the scene. For instance, some of them made a few shots very quick, and after that stopped for a while searching for all the performed changes.

5 Conclusions

In this work, we have created a Virtual Reality shooter game that is designed for making a study in the field of change blindness. Throughout this report, we have discussed the most relevant implementation designs and justified our decisions. Apart from that, we have also tested our game with a number of users, always following an established protocol. The obtained results show that inreasing the cognitive load has influence on the change blindness effect. However, despite this serves as an initial approach in a class project enviroment, we think that further studies would need to be carried out, with a greater amount of experimental subjects, as well as with some more constrains that would allow better and more accurate results. In addition, we have studied some behavioral-related aspects and seen that addressing a higher ammount of tasks also has an influence on subjective sense of users' difficulty and accuracy.

It is clear that for Virtual Reality applications we have to find new ways of studying the attention of users, so these type of studies can help us to understand how do humans pay attention while they are watching a Virtual Reality scene. Having a deep understanding of these factors may help creating more entertaining and immersive applications.

References

[1] George A Alvarez and Patrick Cavanagh. "The capacity of visual short-term memory is set both by visual information load and by number of objects". In: *Psychological science* 15.2 (2004), pp. 106–111.

[2] Katharina Bergmann et al. "Age-related differences in the P3 amplitude in change blindness". In: *Psychological research* 80 (2016), pp. 660–676.

[3] Edurne Bernal-Berdun et al. "SST-Sal: A spherical spatio-temporal approach for saliency prediction in 360 ° videos". In: *Computers Graphics* 106 (2022), pp. 200–209. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2022.06.002. URL: https://www.sciencedirect.com/science/article/pii/S0097849322001042.

[4] Timothy F Brady, Talia Konkle, and George A Alvarez. "A review of visual memory capacity: Beyond individual items and toward structured representations". In: *Journal of vision* 11.5 (2011), pp. 4–4.

[5] Blanca Lasheras-Hernandez, Belen Masia, and Daniel Martin. "DriveRNN: Predicting Drivers' Attention with Deep Recurrent Networks". In: *Spanish Computer Graphics Conference (CEIG)*. Ed. by Jorge Posada and Ana Serrano. The Eurographics Association, 2022. ISBN: 978-3-03868-186-1. DOI: 10.2312/ceig.20221149.

[6] Li-Qian Ma et al. "Change Blindness Images". In: *IEEE Transactions on Visualization and Computer Graphics* 19.11 (2013), pp. 1808–1819.

[7] Daniel Martin, Ana Serrano, and Belen Masia. "Panoramic convolutions for 360° single-image saliency prediction". In: *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*. 2020.

[8] Daniel Martin et al. “A Study of Change Blindness in Immersive Environments”. In: *IEEE Transactions on Visualization and Computer Graphics* 29.5 (2023), pp. 2446–2455.

[9] Daniel Martin et al. “Multimodality in VR: A survey”. In: *ACM Computing Surveys (CSUR)* 54.10s (2022), pp. 1–36.

[10] Daniel Martin et al. “ScanGAN360: A Generative Model of Realistic Scanpaths for 360 Images”. In: *IEEE Transactions on Visualization & Computer Graphics* 01 (2022), pp. 1–1.

[11] *Royalty free stock.* Pixabay. (n.d.) URL: `https://pixabay.com/`. (accessed: 16.05.2023).

[12] Daniel J Simons and Ronald A Rensink. “Change blindness: Past, present, and future”. In: *Trends in cognitive sciences* 9.1 (2005), pp. 16–20.

[13] Evan A Suma et al. “Leveraging change blindness for redirection in virtual environments”. In: *2011 IEEE Virtual Reality Conference*. IEEE. 2011, pp. 159–166.

[14] Madis Vasser, Markus Kängsepp, and Jaan Aru. “Change Blindness in 3D Virtual Reality”. In: *arXiv preprint arXiv:1508.05782* (2015).

A Appendix: Additional resources

A.1 Scripts

We have provided the scripts that define the game logic of the smartphone application. The following scripts can be found in the `./scripts` directory:

- `./scripts/arrowBehavior.cs:` It defines the way in which the arrows get stuck in the cowboy or the floor. It also enable the success and failure sounds.
- `./scripts/changeBlindnessExperiment.cs:` Script that performs the changes in the scene. The possible changes are predefined, and they are performed randomly in runtime. Additionaly, the order of the changes and the number of changes performed is stored.
- `./scripts/gameLogic.cs:` Script that initializes global variables, controls time and loads the game over scene when the time is over.
- `./scripts/gameMovement.cs:` Script that implements the camera movement, the obtaining of the bow and the arrows and the shooting of the arrows.
- `./scripts/gameOverLogics.cs:` Script that is run in the game over scene. It gathers all the player data, and sends it to a Google Forms for a later analysis.
- `./scripts/menuManager.cs:` Script that is run in the initial menu. It loads both the free and experiment game modes, and it also changes the game language.
- `./scripts/targetMovement.cs:` Script that moves the target. It also controls the score of a player during a game.

A.2 Apk

We provide the android application of our game. It can be found in `./apk/western_eina.apk`.

A.3 Demo

We have recorded two short demos of the desktop application:

- `./demo/exploration.mp4:` Demo showing the exploration game mode.
- `./demo/experiment1.mp4:` Demo showing the experiment game mode.
- `./demo/experiment2.mp4:` Demo showing the experiment game mode.

Table 3: Key bindings of the bluetooth controller (vertical mode) for the desktop and smartphone.

Device	Key	Unity code
Smartphone	C	JoystickButton2
Smartphone	B	JoystickButton0
Smartphone	A	JoystickButton3
Smartphone	D	JoystickButton1
Smartphone	Joystick	Axis 5 and 6
Desktop	C	JoystickButton3
Desktop	B	JoystickButton0
Desktop	A	JoystickButton4
Desktop	D	JoystickButton1
Desktop	Joystick	Axis 5 and 6

B Appendix: Bluetooth controller key bindings

We have not found any documentation of the bluetooth controller (vertical mode) that we have used, but we still have managed to configure it. As this information may be very valuable for future projects, we provide the key bindings used for our desktop and android applications. The key bindings for these two devices are different, and we believe that they may vary from one to device to another (although we have not been able to test it). The used key bindings can be found in 3.