GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Fachbereich 12 Informatik und Mathematik
Institut für Informatik

Bachelor Thesis

Distributed Annotation in Virtual Reality

Simon Lööck
Course of Studies: Computer Science

Frankfurt am Main
October 19, 2020

Supervisor: Prof. Dr. Alexander Mehler

Set on October 19, 2020 with LaTeX.

Abstract

The main goal of this work was to create a network environment for the Unity Engine project StolperwegeVR, developed by the Text Technology Lab of Goethe University, in which you will be able to annotate one to several documents in a group. For this, basic network utils like seeing other users or moving objects had to be implemented which had to be easy to use and work with in the future.

Acknowledgements

I want to thank my supervisor Professor Mehler for always supporting me, Giuseppe Abrami and Atilla Kett for helping me when I had difficulties and Melisa Yenal for supporting me and testing my work.

Contents

1 Introduction

StolperwegeVR is an application made with the Unity 3D Engine[1] and developed by the Text Technology Lab of Goethe University in Frankfurt am Main, Germany as a part of the Stolperwege project. It is a Virtual Reality application for experiencing history, tracking the life of specific people during World War 2 along with the original Stolperwege App [8] and amongst other things annotating in Virtual Reality. While there seems to be no connection between these two features, they can be connected for example through text annotation.
Supposing we have two documents of different life sections of a person and we don't know when these sections happened. One report may describe that the person worked at place A and the other may picture his escape to place B. Then we annotate and filter both texts and we discover that the person mentioned place B in the first document, so that it can be suspected he already planned to escape to place B. With this information, the texts can be ordered and we can reconstruct his life.

1.1 Motivation

The primary motivation behind this work was to create a network system for StolperwegeVR that can be used for future work at the application. Additionally, since the old possibility for distributed annotation was just working at a browser, a more motivating solution was desirable. After a lot of past experience, using a Virtual Reality application is very motivating and applicable for text annotation since Attila Kett already implemented the required tools for this.

1.2 Related Work

So far there were a few applications which provided network-compatible Virtual Reality functions, for example, VRChat [17] where you are able to connect with anyone with and without a Virtual Reality headset in so called worlds. In these worlds you can interact with different objects and other players. Furthermore you can change the look of the avatar to many different persons and creatures, mostly provided by the community.
Many other applications have similar features, but most of them have primarily humorous purposes and - according to my research - none of them has distributed text annotating functionalities.
The best previous solution of the Text Technology Lab for distributed text annotation

[1] `https://unity3d.com/de/unity`

was to use the TextAnnotator [3][5] Website[2] along with third-party applications for communication. The TextAnnotator is a browser-based framework for annotation purposes like part of speech tagging, named entity recognition and so on. It uses a UIMA Database Interface [1] to process database-based UIMA documents. All of this should be improved with this work for easy distributed text annotation.

1.3 Structure

The work was split into three phases. The first phase was to know and analyze existing components that could be useful. The second phase was to implement everything that was needed for a good working network experience, along with the Annotator Tool. For this, many basic network components had to be implemented since they were either missing or not working. The last phase was to evaluate the outcome and analyze whether it was better than the pre-existing system.

1.4 Notation

When addressing coordinates, the Unity 3D Engine standard [16] will be used:

- x: right
- y: up
- z: forward

[2]`http://www.textannotator.texttechnologylab.org`

2 Applicable Components

Some useful elements were already existing that could be used for this work. Mainly the network components and the Annotator Tool were beneficial.

2.1 Virtual Rooms

VirtualRooms are the scenes of StolperwegeVR which are dedicated to a networked room. Any user with necessary permissions can host and connect to a VirtualRoom. The specifications of the VirtualRooms, as well as the permissions are stored in the Stolperwege database.
Currently existing permissions are:

0. None
1. Read
2. Write
3. Delete
4. Grant

Apart from that, a VirtualRoom can be public, so that (currently) every user has permission class 3 (Delete). Every permission inherits from the lower permissions, for instance, permission class 4 extends class 0 to 3.
Any user that has permission class 1 (Read) for a VirtualRoom can excess it, class 2 (Write) can change content and class 3 (Delete) can add and remove content, which had to be implemented.
The VirtualRooms were used as a base for all new network components, while it was ensured that only users with the right permission can contribute.

2.2 Network Connectivity

Fundamental network connectivity tools were previously implemented with a system to choose between VirtualRooms and connecting to them. Until then, only the connection was successful, it was neither possible to see another user trouble-free nor to interact with anything.
It uses the Unity build-in network function UNET [11] which could be used for further work. In the end, only a part of it was used, since on the one hand, it did not hold everything necessary for this work and on the other hand, it was already deprecated, so that if the engine version of StolperwegeVR will be updated, most algorithms could be used again.

2.3 Annotator Tool

The Annotator Tool is the tool for annotating a text in StolperwegeVR developed by Attila Kett as a component of VAnnotatoR [9][10][2]. The VAnnotatoR is a part of StolperwegeVR, in which you are able to annotate visualizations of, for example, individual words of documents, images, videos, and sound files jointly.
The Annotator Tool connects to the web-service of the TextAnnotator and can open a document along with the tools to annotate it. The Annotator Tool updates itself whenever the annotation of the text is changed and displays it through token objects with different colors.
The previous use of the Annotator Tool was in a specific room in the Resources2City Explorer [6][6], where a document is transformed into a generated city. The room contains everything necessary to annotate a document. The tool was quite useful for this work as it already held nearly all necessities for distributed annotating.
Since it is connected to the web-service of the TextAnnotator, it synchronizes every change of the document immediately regardless of the origin.
Related work to the VAnnotatoR that is worth mentioning is the gesture-based interface in StolperwegeVR [7]. With this work and the gesture-based interface, it could be possible to write and annotate texts simultaneously in Virtual Reality in the future, only using gestures without any kind of virtual keyboard.
Additionally,

3 Tasks

Since the only available network component was the connectivity between users in a VirtualRoom, some basic tools had to be implemented first. After it was possible to interact with other users and objects, the components for distributed text annotations could be developed.

3.1 Basic Network Tools

For reasons that will be explained in section 3.2, the Network had to be partly constructed client-based. For this, the following two new components, NetworkClientTransform and CustomNetworkIdentity, were implemented. Additionally the component NetworkComponentIdentifier was implemented which assigns every component of a GameObject an identification number so that if data is sent and the GameObject has multiple components that could recieve the data, the right one is chosen (see fig. 3.1).

3.1.1 CustomNetworkIdentity

This class is the equivalent to the UNET component NetworkIdentity [12]. It gives every networked GameObject a unique ID through which it can be identified on every client, so that information is sent to the correct GameObject.
In this case, the ID is determined by the ID of the user that spawned the object along with a counter that increases every time a new ID is used. The GameObject will be then saved in a Dictionary with the ID as a key so that if a client receives information, the correct GameObject will be chosen.
This component is used for custom generated GameObjects like DrawedObjects, which can be drawn inside StolperwegeVR. Those objects cannot be spawned with the conventional method since it demands the registration of a GameObject in order to spawn it on all clients. How those drawn objects can be spawned nonetheless will be explained in section 3.1.3.

3.1.2 NetworkClientTransform

This component is the pendant to the UNET component NetworkTransform [13]. It synchronizes the position, rotation, and local scale of a GameObject on all clients. The reason why this component needed to be implemented instead of using the NetworkTransform is that the client cannot interact with Server objects the way it was necessary.

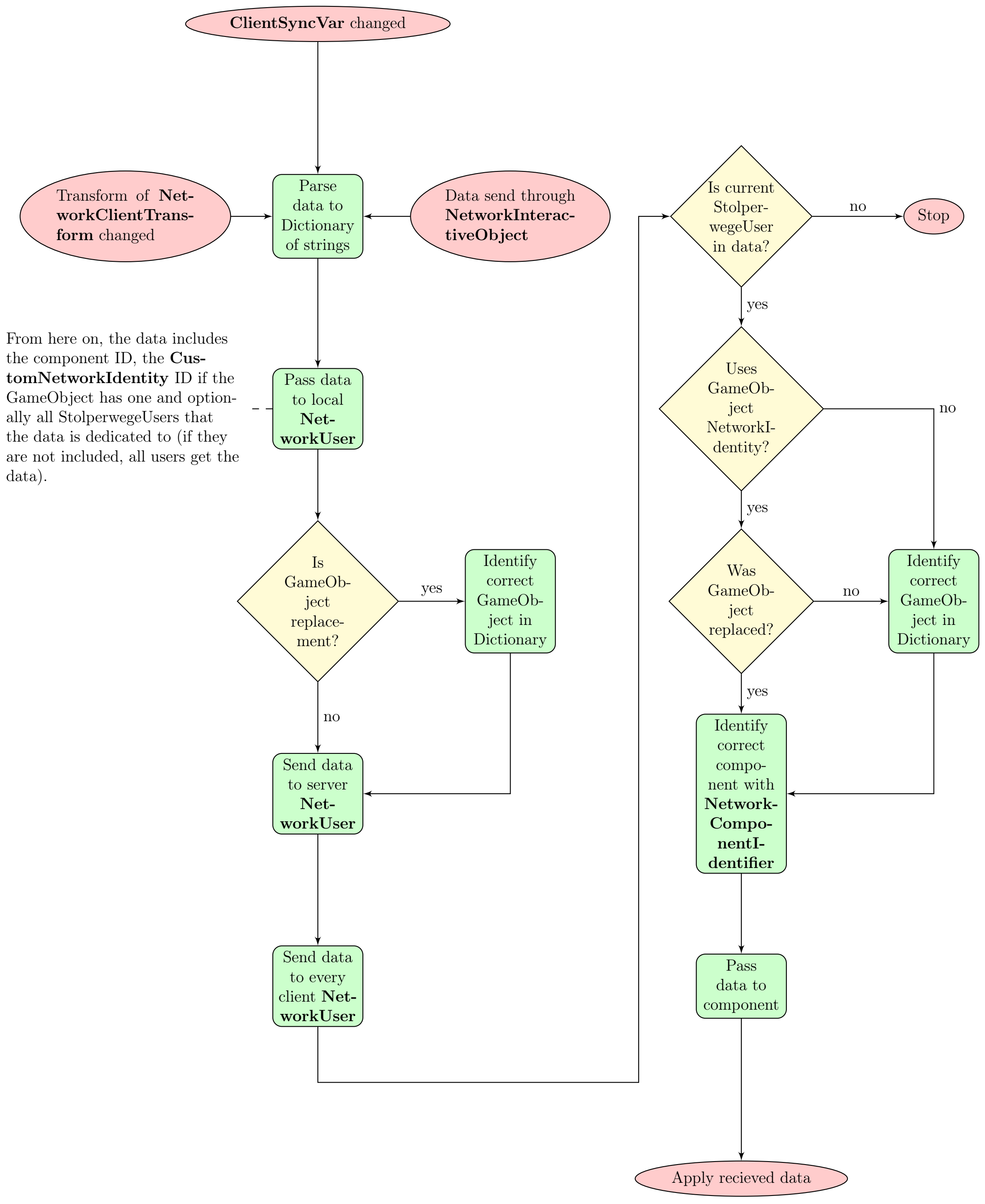

ClientSyncVar changed
Transform of NetworkClientTransform changed
Parse data to Dictionary of strings
Data send through NetworkInteractiveObject
From here on, the data includes the component ID, the CustomNetworkIdentity ID if the GameObject has one and optionally all StolperwegeUsers that the data is dedicated to (if they are not included, all users get the data).
Pass data to local NetworkUser
Is GameObject replacement?
yes
Identify correct GameObject in Dictionary
no
Send data to server NetworkUser
Send data to every client NetworkUser
Is current StolperwegeUser in data?
no
Stop
yes
Uses GameObject NetworkIdentity?
no
yes
Was GameObject replaced?
no
Identify correct GameObject in Dictionary
yes
Identify correct component with NetworkComponentIdentifier
Pass data to component
Apply recieved data

Figure 3.1: How NetworkInteractiveObjects and NetworkClientTransform communicate with the equivalents on the clients

3.1.3 NetworkInteractiveObject

This class is an extension of the InteractiveObject. The InteractiveObject is a component to make a GameObject interactive in StolperwegeVR. Amongst other things, it can be grabbed and clicked.
The NetworkInteractiveObject additionally has the ability to send and receive data. It adds the NetworkClientTransform, and if needed the CustomNetworkIdentity, automatically and spawns and synchronizes the object on all clients. The inheritance of the InteractiveObject was chosen to simplify future work since on the one hand the ability to grab and to click a GameObject can be turned off and on the other hand, it is the most used class in this project. It is much easier and more efficient to create the ClientSyncVar for this particular component.
The ClientSyncVar can be compared to the SyncVar of UNET [15], however, it synchronizes on each client if anyone changes it on the client-side. At the moment not all types are supported, but the support can be implemented rather quickly in the NetworkInteractiveObject.

3.1.4 Spawning GameObjects

There are two ways of spawning GameObjects on the server:

1. Before runtime: Add a NetworkIdentity to the GameObject, register it as spawnable Prefab on the NetworkManager, add a NetworkInteractiveObject and set the string ObjectPath as the path from the Resources folder to the prefab

2. Before or during runtime: Do not add a NetworkIdentity, add a NetworkInteractiveObject and set the variable ColliderType to the Type of the wanted collider.

When using option 1, a message is sent to the server that a prefab at ObjectPath wants to be instantiated. The prefab will be spawned on the server and the client that created it will create a forwarding link from the spawned to the original object so that it seems like it is the same. Additionally, all MeshRenderers and Colliders will be destroyed so that it cannot be seen or used by the client that created the object. Instead, he or she can interact with the original, and all other objects will act the same.
When using option 2, the mesh of the GameObject will be sent through the network that will be reconstructed on all clients. This is not recommended for complex meshes since there will be a lot of data to send.

3.2 Multiple Users in a VirtualRoom

One of the biggest challenges was the NetworkUser. Previous attempts to make the default StolperwegeVR user object network capable failed because it was too complex. Two users could connect but one user controlled the other and none of them

could move. Apart from that, there would have been a lot more problems for input, InteractiveObjects and so on. Imagine having your Virtual Reality headset on when suddenly your connection breaks down and your own sight lags. This could result for example in motion sickness and frustration.
The solution was simple: to make a separate empty NetworkUser object which spawns the default StolperwegeVR user if it represents the local user and an avatar (see fig. 3.2) else. This worked well but also added some problems. Since the StolperwegeVR user does not exist on the server, the actions could not be caught on the server-side. Because of that, the whole network system had to be construct partly client-based, meaning a client can change something on his local machine and it will be synchronized on all other clients. This could add a bit of delay since the client has to send all the data to the server which then sends it to all clients instead of just sending input commands to the server which then reacts to it and sends the result to all clients. But it can provide a more enjoyable experience since for example the only problems with the Virtual Reality headset will be on the local machine and cannot caused by the network.
By virtue of UNET, the NetworkUser was used as a tunnel for all messages that will be sent through the network using ClientRpc [14] and Command [14]. Through these functions, dictionaries containing string keys and values can be sent to the corresponding GameObject of all clients or specific StolperwegeUsers. The function sorts and assigns the information to the correct GameObject and component which then can react to it.
The NetworkUser synchronizes position, y-rotation, and height of the user as well as the position and rotation of the hands through NetworkTransform components of UNET via empty GameObjects that follow the movements of the Virtual Reality user. In addition to that, the state of the hand (relaxed, pointing, grabbing, and fist) and the StolperwegeUser of the client is synchronized by using the functions mentioned above. Along with the allocation of the StolperwegeUser, the avatar will be painted using the specifications made by the StolperwegeUser that are saved in the database.
Thereby users can see and interact with each other without creating problems with the controls together with the advantage of not having to spawn a complex user object every time a user connects. An example result of the StolperwegeUsers dummy and abrami can be seen in fig. 3.2. Note that none of the users had avatar color specifications at that moment.

3.3 Communication between Users

Since the main goal was to create a working space in Virtual Reality where users can work together, some sort of communication was reasonable. There were three options:

1. Text chat with text input (combinable with text to speech)
2. Text chat with voice input (speech to text, also combinable with text to speech)

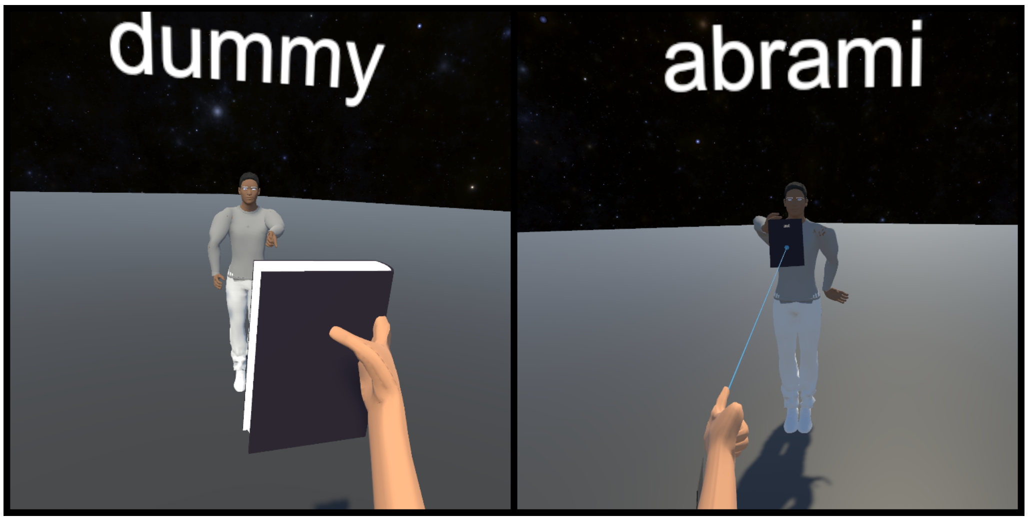


Figure 3.2: How users see other users and network objects

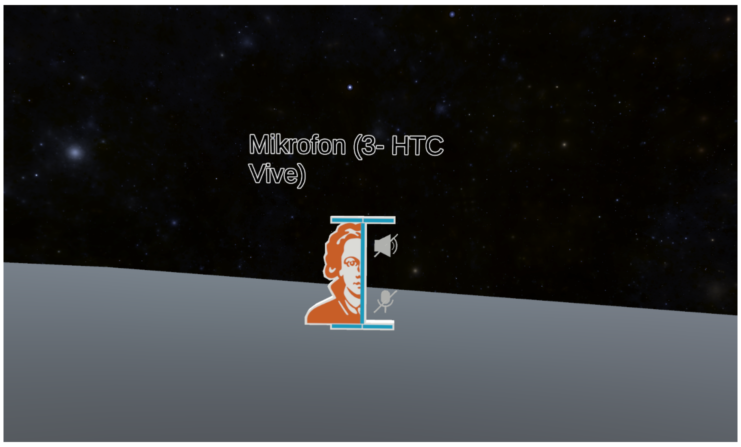


Figure 3.3: A turned off communicator

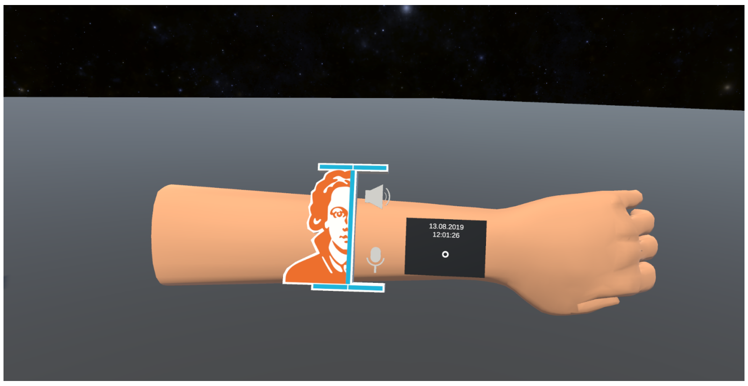


Figure 3.4: a turned on communicator

3. Voice chat

The one option that will be neglected is the usage of third party applications. Since the goal was to create a complete working system for distributed working, the possibility to communicate had to be implemented.
Option one was rejected since it would be too overelaborate to type something on a keyboard in Virtual Reality every time you want to communicate. Apart from that, there is some kind of natural text chat through the text type of an InfoSurface which was also made network compatible with which you can write messages that all other users can read on the InfoSurface.
Option two could have been promising, but it would have been too complicated, and since it was not the central goal, it was also rejected.
At last, option three was chosen. It was implemented using Unity's Microphone class. The AudioInput script captures when turned on a 0.2 seconds long clip of the selected microphone, converts the clip to a float stream of values between -1 and 1 which is checked for data worth sending and then send to all other clients. Data is worth sending if the absolute value is bigger than 0.05. Everything else will be interpreted as silence. The received data is then transferred to the current AudioOutput script, converted into a clip which will be played.
The controls, which microphone device is used and if the AudioInput is turned on, lies within the Communicator GameObject. When accessing a VirtualRoom, one Communicator for each device will be generated (plus one if you have no microphone so that you can at least hear the other users) in the bottom part of a shelf as you can see in fig. 3.9. At the start, each Communicator will be turned off (see fig. 3.3) and can be turned on by clicking on it. If you do so, the Communicator will be stuck to your left hand above the clock for easy access (see fig. 3.4). By clicking on it again, the AudioInput will be turned off so that you are muted. As long as you have a Communicator on the arm, you will be able to hear all users that have turned on the AudioInput. If you grab the communicator it will come off your arm and you won't be able to hear anyone anymore and no one will be capable of hearing you.

3.4 Network-capable Annotator Tool

Since the TextAnnotator already synchronizes the annotations, making it network capable was quite simple. The first step was to make the existing annotator portable since the original was designed to be a table in a room. Since the goal was to annotate as many documents as needed with as many users as wanted, there had to be a possibility to create multiple instances and place them as wished. The result of the modifications can be seen on the left side of fig. 3.6. To instantiate a portable annotator, you have to create an InfoSurface, choose a document instance (see fig. 3.5), insert a resource from the ResourceManager and click on the "Annotate" button at the bottom of the IOEditor that opens if you long click on the InfoSurface that is seen on the right side of fig. 3.6.
The portable annotator uses a NetworkInteractiveObject so that all transformations will be synchronized, as well as the loaded document and sentence pointer.

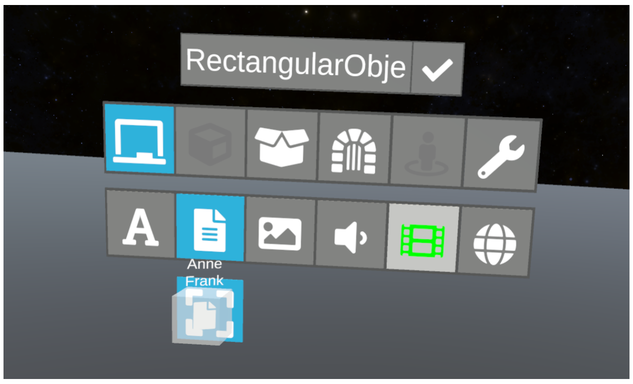
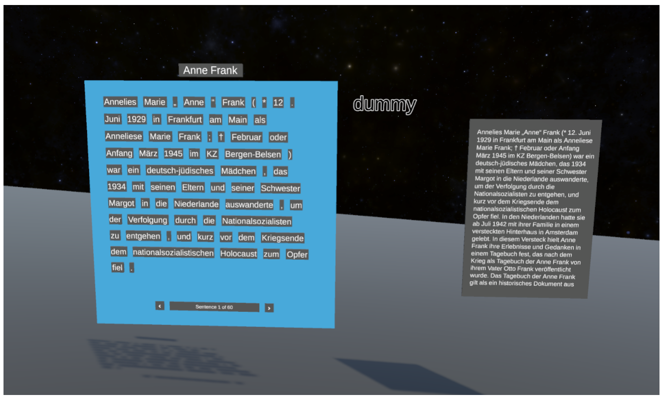


Figure 3.5: Where to insert a ResourceData object in an InfoSurface of document type



Anne Frank

Annelies Marie „ Anne “ Frank (* 12 . Juni 1929 in Frankfurt am Main als Anneliese Marie Frank ; † Februar oder Anfang März 1945 im KZ Bergen-Belsen) war ein deutsch-jüdisches Mädchen , das 1934 mit seinen Eltern und seiner Schwester Margot in die Niederlande auswanderte , um der Verfolgung durch die Nationalsozialisten zu entgehen , und kurz vor dem Kriegsende dem nationalsozialistischen Holocaust zum Opfer fiel .

Sentence 1 of 60

Annelies Marie „Anne“ Frank (* 12. Juni 1929 in Frankfurt am Main als Anneliese Marie Frank; † Februar oder Anfang März 1945 im KZ Bergen-Belsen) war ein deutsch-jüdisches Mädchen, das 1934 mit seinen Eltern und seiner Schwester Margot in die Niederlande auswanderte, um der Verfolgung durch die Nationalsozialisten zu entgehen, und kurz vor dem Kriegsende dem nationalsozialistischen Holocaust zum Opfer fiel. In den Niederlanden hatte sie ab Juli 1942 mit ihrer Familie in einem versteckten Hinterhaus in Amsterdam gelebt. In diesem Versteck hielt Anne Frank ihre Erlebnisse und Gedanken in einem Tagebuch fest, das nach dem Krieg als Tagebuch der Anne Frank von ihrem Vater Otto Frank veröffentlicht wurde. Das Tagebuch der Anne Frank gilt als ein historisches Dokument aus

Figure 3.6: Left: A network capable portable annotator; Right: The parent InfoSurface of the annotator

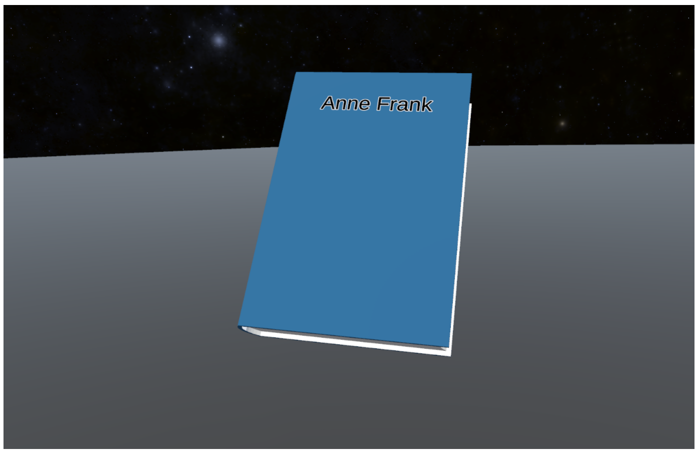


Figure 3.7: A closed book that holds data with the title "Anne Frank"

3.5 Permissions for VirtualRooms and Annotator Tool

As described in section 2.1, there are currently five permission types for the StolperwegeVR. It was already implemented that only users with permission class 1 (Read) can enter the specified VirtualRoom and only users with permission class 4 (Grant) can grant and revoke permissions for the VirtualRoom. The following was additionally implemented:
Permission class 2 (Write)
Users with the permission to write in a VirtualRoom can grab and click on NetworkInteractiveObjects that were spawned on the server. When they try to instantiate a NetworkInteractiveObject, it will not be spawned on the server but exists on the client who is able to interact with it.
Permission class 3 (Delete)
Users with the permission to delete in a VirtualRoom are able to add and remove NetworkInteractiveObjects. Every NetworkInteractiveObject that the instantiate will be spawned on the server so that every client can see it.

The permissions for the Annotator Tool are managed through the TextAnnotator. As soon as a document is opened on an InfoSurface or a portable annotator, the user has to log into a TextAnnotator account. This account determines if you can open and edit the specific document.

3.6 Distributed working on one to several Annotation Tasks in a VirtualRoom

The final task was to bring it all together to create an environment in which multiple users can annotate several documents.

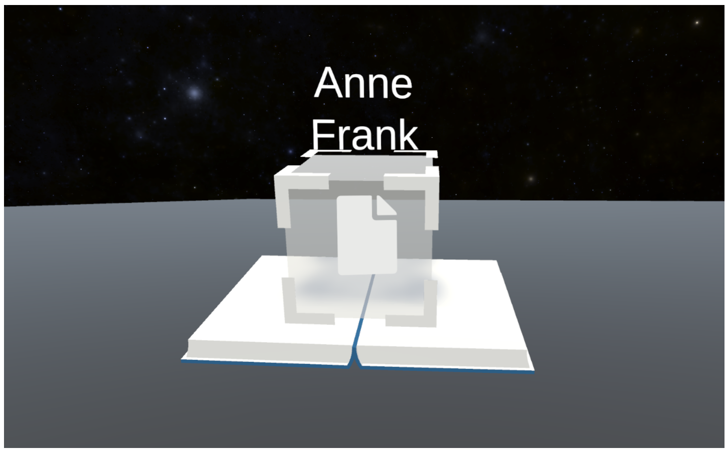


Figure 3.8: The same book as in fig. 3.7 but opened. You can see the ResourceData object it holds

Since the portable annotators use data from the RESOURCEMANAGER [4], which already can be visualized as a DataResource, it would have been valuable if you could share and save this data. Making the existing visualizations network capable would not have been very useful since once you use them, for example, for an InfoSurface, they disappear. So a container was constructed in which you can store and replicate any DataResource. These containers appear as books as seen in fig. 3.7 since they stand for securing and distributing information. The title of the book matches the enclosed DataResource, which can be inserted when the book is opened as seen in fig. 3.8. To replicate the contained information, all you have to do is long click on the DataResource that is seen when you open the book. The books will be spawned with a random size and color on the shelf as seen in fig. 3.9, and the data can always be overwritten by users that have the permission to write in the VirtualRoom.
The final problem was that the TEXTANNOTATOR is only capable of opening a connection to one document at a time. Since there should be the possibility to annotate on multiple documents at once, the other annotation processes would not have worked anymore once another is opened. The solution was simple. There is always only one active portable annotator on a client, all others are paused. They do not receive or send any annotation updates as long as it is not unpaused, which can be done effortlessly by clicking on the pause screen (see fig. 3.10). Each client can have a different active annotator so that they will not be limited. As soon as you unpause a portable annotator, the name of the current StolperwegUser will be displayed on the right side of the annotator (see fig. 3.6) on all clients so that everyone knows which annotator you are currently using.

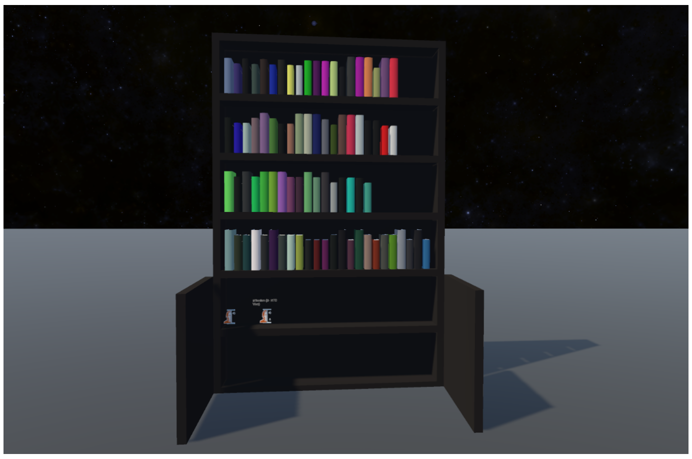
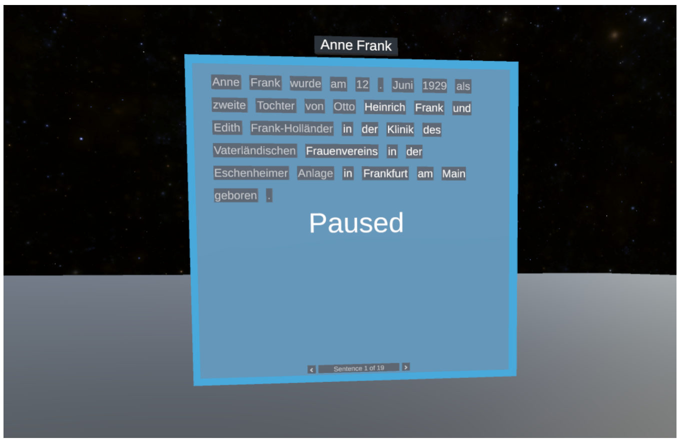


Figure 3.9: The shelf that exists in every VirtualRoom that holds empty books and the communicators



Anne Frank wurde am 12 . Juni 1929 als zweite Tochter von Otto Heinrich Frank und Edith Frank-Holländer in der Klinik des Vaterländischen Frauenvereins in der Eschenheimer Anlage in Frankfurt am Main geboren .

Figure 3.10: A paused portable annotator

4 Evaluation

To test the efficiency of the new system in comparison to the web-interface, an evaluation had to be carried out.

4.1 Structure

The evaluation was structured in two parts. The first part was the testing of the old solution, using the TextAnnotator website[1], a small self-made Java application for recording the results and Discord[2] for communication. The second part was the testing of the new solution, the Annotator Tool in a Virtual Room in StolpersteinewegeVR.
Both parts had the same tasks:

1. trying to communicate without any Chat-like feature and
2. discussing over Voice-chat.

In both tasks, the participants had to annotate a specific type of words in four sentences of a document. Additionally, only one participant knew which type of words had to be annotated in task one, and he had to try to explain it to the other. The attendees were given two Annotator Tools in the VirtualRoom so that they had the possibility to divide the second task since both knew what they had to do.
Subsequently, the participants had to fill in a short questionnaire. The original questionnaire in German can be seen in the appendix.

4.2 Results

The evaluation was carried out with twelve pairs, where all uneven pairs started with the TextAnnotator in the Browser, and all others started with the Annotator Tool in a Virtual Room. Task 1 was correct if the second participant guessed correct which type of words he had to annotate. The results can be seen in table 4.3.

[1] `http://www.textannotator.texttechnologylab.org`
[2] `https://discordapp.com/`

	Task 1		Task 2
Pair	Time (in sec)	Correct	Time (in sec)
1	219.8	0	277.2
2	255.8	0	404.8
3	368.3	1	293.0
4	183.5	1	257.9
5	279.0	1	429.6
6	294.3	1	332.5
7	513.3	1	232.2
8	232.5	1	159.6
9	349.1	1	450.3
10	414.4	1	546.6
11	127.2	0	549.5
12	286.5	1	219.9

Table 4.1: Results of the TextAnnotator part

	Task 1		Task 2
Pair	Time (in sec)	Correct	Time (in sec)
1	343.4	1	405.9
2	757.4	0	117.1
3	315.3	0	454.7
4	353.4	1	293.7
5	347.9	1	418.8
6	827.0	0	480.7
7	358.7	1	259.3
8	474.1	1	336.8
9	483.1	1	749.8
10	856.1	0	213.3
11	421.7	1	512.8
12	616.1	1	174.5

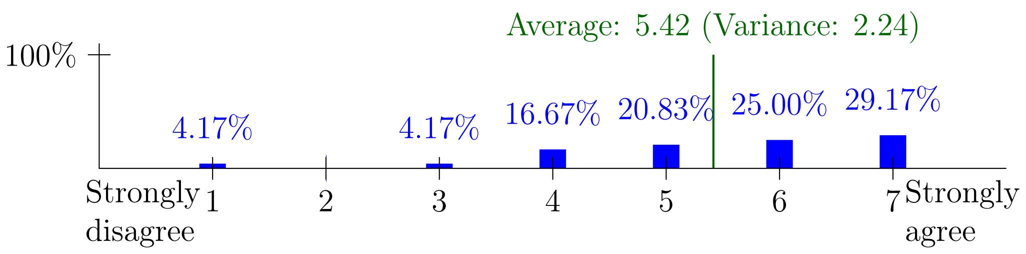
Table 4.2: Results of the Annotator Tool part

	Task 1		Task 2
Part	Time (in sec)	Correct	Time (in sec)
TextAnnotator	293.6	9/12	346.1
Annotator Tool	512.8	8/12	368.1

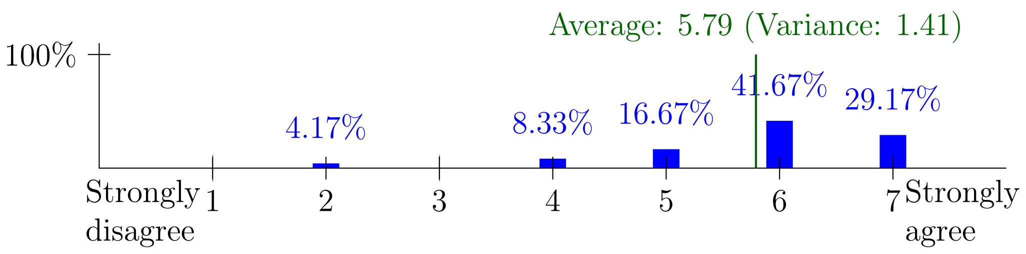
Table 4.3: Average Results

The results of the questionnaire are the following:

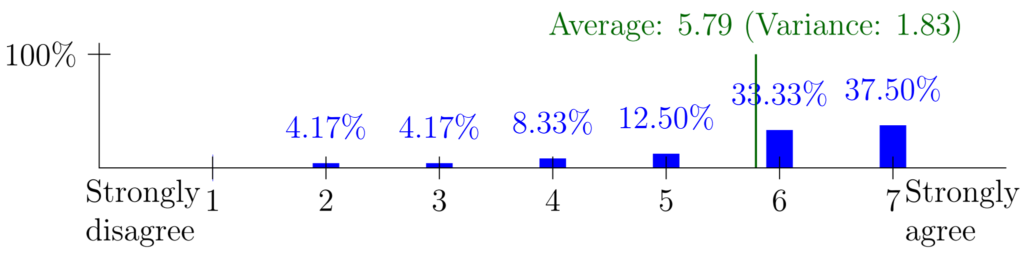
1. I am satisfied with the way the communicators work



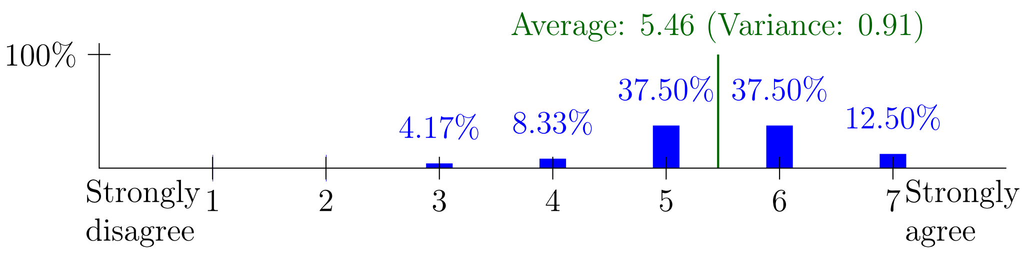
2. I am satisfied with the way I see the movements of other users (apart from the look of the avatar)



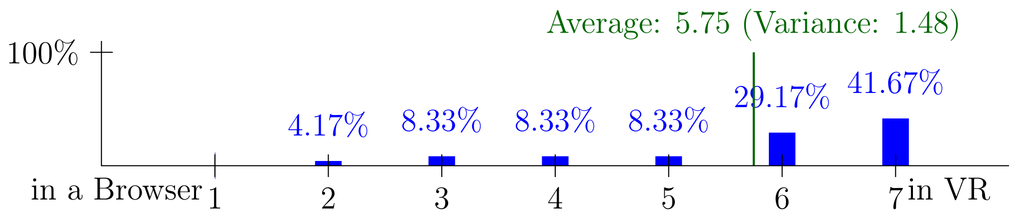
3. The annotation of texts with several people at the same time has increased my motivation.



4. The annotation tool meets my expectations for distributed annotation.



5. I am more motivated to annotate a text with a colleague or in a group in this environment



What stood out during the evaluation was the confusion about the sentence numbering. Since the web-interface displays parts of a document without paying attention to the sentences, most participants miscounted the correct numbering, whereas the Annotator Tool only displays one sentence of a document at a time. Along with the synchronization of the current sentence position, this helped many participants finding the correct sentences.
Since the main focus was on the distributed aspect of the new system, the correctness of the annotations weren't checked. But according to my observations during the evaluation, it was more or less equal.

5 Conclusion

5.1 Results of the Evaluation

Since there were only twelve pairs involved in the evaluation, the results in table 4.3 can only be used partially.
Apart from the average times of task 1, the results are more or less identical. The time difference could be explained with the initial confusion about the handling in Virtual Reality since most attendees experienced it for the first time. To solve the first task, almost all participants used the same technique of annotating the words themselves so that their partner had to guess which was the correct category. To solve the second part in Virtual Reality, some used both Annotators to be more efficiently. Nonetheless, most participants remarked that the distributed Virtual Reality method of annotating a text was far more motivating than distributed annotating in the web-interface. This review is also reflected in the results of the questionnaire. The distribution of the answers to the first question was virtually expected since the communicators work rather simple and are not the ideal way of implementing a voice-chat.
All in all, the result of this work can be called successful since the main goal was to create a stable working environment for motivating distributed text annotating, which most participants of the evaluation experienced.

5.2 Future Work

As already mentioned, the voice chat is currently a bit rudimentary. Making a fully working chat would have been too complicated for this work since the focus was on the distributed text annotation possibility additionally to the working networkability of StolperwegeVR. Adding a better solution for the voice-chat could be a task for future students.
Additionally, some other opportunities were discovered to improve the experience. For example, the Annotator Tool could be renewed for smoother annotations. Also, the Avatar has to be redone since some bugs were discovered like the vanishing of some parts when the user is looking at it from different angles.
At last, the network usability of the StolperwegeVR application can be expanded more efficiently since many components that were implemented for this work can be used simply. For example, any InteractiveObject can be made network-compatible by changing it to a NetworkInteractiveObject and making sure it will be spawned (see section 3.1.4). Furthermore, ClientSyncVars (see section 3.1.3) can be utilized to synchronize any variable with a value or an implemented type.

Bibliography

[1] Giuseppe Abrami and Alexander Mehler. “A UIMA Database Interface for Managing NLP-related Text Annotations”. In: *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12.* LREC 2018. Miyazaki, Japan, 2018.

[2] Giuseppe Abrami, Alexander Mehler, and Christian Spiekermann. “Graph-based Format for Modeling Multimodal Annotations in Virtual Reality by Means of VAnnotatoR”. In: *Proceedings of the 21th International Conference on Human-Computer Interaction, HCII 2019.* Ed. by Constantine Stephanidis and Margherita Antona. HCII 2019. Orlando, Florida, USA: Springer International Publishing, July 2019, pp. 351–358. ISBN: 978-3-030-30712-7.

[3] Giuseppe Abrami et al. “TextAnnotator: A flexible framework for semantic annotations”. In: *Proceedings of the Fifteenth Joint ACL - ISO Workshop on Interoperable Semantic Annotation, (ISA-15).* ISA-15. accepted. Gothenburg, Sweden, May 2019.

[4] Rüdiger Gleim, Alexander Mehler, and Alexandra Ernst. “SOA implementation of the eHumanities Desktop”. In: *Proceedings of the Workshop on Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts, Digital Humanities 2012.* Hamburg, Germany, 2012.

[5] Philipp Helfrich et al. “TreeAnnotator: Versatile Visual Annotation of Hierarchical Text Relations”. In: *Proceedings of the 11th edition of the Language Resources and Evaluation Conference, May 7 - 12.* LREC 2018. Miyazaki, Japan, 2018.

[6] Attila Kett et al. “Resources2City Explorer: A System for Generating Interactive Walkable Virtual Cities out of File Systems”. In: *Proceedings of the 31st ACM User Interface Software and Technology Symposium.* Berlin, Germany, 2018.

[7] Vincent Roy Kühn. *A gesture-based interface to VR.* Bachelor Thesis. Bachelor. Goethe University of Frankfurt. 2018. URL: `http://publikationen.ub.uni-frankfurt.de/frontdoor/index/index/docId/50915`.

[8] Alexander Mehler et al. “Stolperwege: An App for a Digital Public History of the Holocaust”. In: *Proceedings of the 28th ACM Conference on Hypertext and Social Media.* HT ’17. Prague, Czech Republic: ACM, 2017, pp. 319–320. ISBN: 978-1-4503-4708-2. DOI: `10.1145/3078714.3078748`. URL: `http://doi.acm.org/10.1145/3078714.3078748`.

[9] Alexander Mehler et al. “VAnnotatoR: A Framework for Generating Multimodal Hypertexts”. In: *Proceedings of the 29th ACM Conference on Hypertext and Social Media.* Proceedings of the 29th ACM Conference on Hypertext and Social Media (HT ’18). Baltimore, Maryland: ACM, 2018.

[10] Christian Spiekermann, Giuseppe Abrami, and Alexander Mehler. "VAnnotatoR: a Gesture-driven Annotation Framework for Linguistic and Multimodal Annotation". In: *Proceedings of the Annotation, Recognition and Evaluation of Actions (AREA 2018) Workshop*. AREA. Miyazaki, Japan, 2018.

[11] *Unity - Manual: Multiplayer and Networking*. Apr. 15, 2019. URL: `https://docs.unity3d.com/2018.3/Documentation/Manual/UNet.html` (visited on 09/13/2019).

[12] *Unity - Manual: Network Identity*. Apr. 15, 2019. URL: `https://docs.unity3d.com/2018.3/Documentation/Manual/class-NetworkIdentity.html` (visited on 09/13/2019).

[13] *Unity - Manual: Network Transform*. Apr. 15, 2019. URL: `https://docs.unity3d.com/2018.3/Documentation/Manual/class-NetworkTransform.html` (visited on 09/13/2019).

[14] *Unity - Manual: Remote Actions*. Apr. 15, 2019. URL: `https://docs.unity3d.com/2018.3/Documentation/Manual/UNetActions.html` (visited on 09/13/2019).

[15] *Unity - Manual: State synchronization*. Apr. 15, 2019. URL: `https://docs.unity3d.com/2018.3/Documentation/Manual/UNetStateSync.html` (visited on 09/13/2019).

[16] *Unity - Scripting API: Vector3*. Apr. 15, 2019. URL: `https://docs.unity3d.com/2018.3/Documentation/ScriptReference/Vector3.html` (visited on 09/22/2019).

[17] *VRChat*. 2019. URL: `https://www.vrchat.net/` (visited on 09/13/2019).

Appendices

Simon Lööck
E-Mail: simon.loeoeck@web.de

Nutzerstudie zu verteilten Annotationen in virtuellen Umgebungen

UMUX-Fragebogen

12. September 2019

Zur Auswertung des verteilten Annotationstools wird die *Usability Metric for User Experience* (UMUX, Finstad 2010) verwendet. UMUX umfasst die folgenden fünf Fragen, die Sie bitte per Ankreuzen jeweils eines Feldes beantworten.

1. Ich bin zufrieden mit der Funktionsweise der Kommunikatoren

1	2	3	4	5	6	7
Stimme nicht zu						Stimme voll und ganz zu

2. Ich bin zufrieden, wie ich die Bewegungen anderer User sehen kann (abgesehen vom Aussehen des Avatars)

1	2	3	4	5	6	7
Stimme nicht zu						Stimme voll und ganz zu

[bitte wenden]

3. Die Annotation von Texten mit mehreren Personen gleichzeitig hat meine Motivation gesteigert.

1	2	3	4	5	6	7
Stimme nicht zu						Stimme voll und ganz zu

4. Das Annotations-Werkzeug erfüllt meine Erwartungen an das gemeinsame Annotieren.

1	2	3	4	5	6	7
Stimme nicht zu						Stimme voll und ganz zu

5. Ich bin eher motiviert mit einem Kollegen oder in einer Gruppe hier einen Text zu annotieren

1	2	3	4	5	6	7
im Browser						in VR

6. Anmerkungen (Optional)

Literatur

Finstad, Kraig (2010). „The usability metric for user experience“. In: *Interacting with Computers* 22.5, S. 323–327.

Bitte dieses Formular zusammen mit der Abschlussarbeit abgeben!

Erklärung zur Abschlussarbeit

gemäß § 25, Abs. 11 der Ordnung für den Bachelorstudiengang Informatik vom 06. Dezember 2010:

Hiermit erkläre ich Herr / Frau

____

Die vorliegende Arbeit habe ich selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst.

Frankfurt am Main, den

____

Unterschrift der Studentin / des Studenten