

Collaborative Software Exploration with the HTC Vive in ExplorViz

Bachelor's Thesis

Malte Hansen

September 29, 2018

KIEL UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
SOFTWARE ENGINEERING GROUP

Advised by: Prof. Dr. Wilhelm Hasselbring
M.Sc. Christian Zirkelbach

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Kiel, 29. September 2018

Abstract

The complexity of software is constantly increasing and therefore understanding software can be challenging. Especially for projects in which many developers are involved, it is of great importance to use technologies for the acquisition and transfer of knowledge of software. Visualizations of software landscapes are one possible approach for this task. However, with the help of visualization alone, exploring unknown software is still a potentially challenging and time-consuming task.

In this thesis we present a new approach using virtual reality for exploring software landscapes collaboratively. We are building on top of the research tool ExplorViz, which provides technologies to visualize software landscapes. We use technologies like the HTC Vive to allow the collaborative exploration of software in virtual reality. ExplorViz is a Web-based application and thus our approach is mostly platform independent. With the use of a microservice architecture and WebSocket connections to exchange data we strive to achieve both modular expandability and good performance for the real-time user environment.

As a proof of concept, we conducted a usability evaluation with 22 probands. The results of this evaluation indicate for a good usability. Additionally, the user feedback gives indications for further improvements and paves the way for a future study to evaluate our approach in comparison to other ways of exploring software collaboratively.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Goals	2
1.3	Document Structure	3
2	Foundations and Technologies	5
2.1	Virtual Reality	5
2.2	HTC Vive	5
2.3	ExplorViz	6
2.4	ExplorViz VR extension	7
3	Concept	11
3.1	Architecture	11
3.2	Backend extension	12
3.3	Frontend extension	13
4	Implementation	17
4.1	Backend Implementation	17
4.1.1	Connection to Frontend Extension	17
4.1.2	Synchronization	18
4.2	Frontend Implementation	18
4.2.1	Connection to Backend Extension	18
4.2.2	Main Loop	19
4.2.3	Landscape	20
4.2.4	Applications	22
4.2.5	User Representation	23
4.2.6	Menu	25
4.2.7	Text messages	26
4.2.8	Teleportation & Height Adjustment	26
4.2.9	Spectating	27
4.2.10	Further Adjustments	28
5	Evaluation	29
5.1	Goals	29
5.2	Questionnaire	30
5.3	Experimental Setup	33

Contents

5.4	Execution of the Experiment	34
5.5	Results	36
5.6	Discussion	39
5.7	Threats to Validity	41
5.8	Summary	41
6	Related Work	43
7	Conclusions and Future Work	45
7.1	Conclusions	45
7.2	Future Work	45
	Bibliography	47
	Appendix A	49
	Appendix B	57
	Appendix C	61

Introduction

The complexity of software is constantly increasing and therefore understanding software can be challenging. Especially for projects in which many developers are involved, it is of great importance to use technologies for the acquisition and transfer of knowledge about software. Visualizations of software landscapes are one possible approach to this task. ExplorViz¹ is a software for exploring and understanding software landscapes which has been under development since 2012 [Fittkau et al. 2013]. ExplorViz can be helpful to familiarize oneself with a software and is suitable to understand the architecture and data streams. Already in its early development, there has been an effort to incorporate advanced technologies into ExplorViz. The goal is to provide new and diverse opportunities to experience and explore software.

A promising technology for this is virtual reality (VR). Firstly, VR was introduced to ExplorViz using the Oculus Rift DK1 [Fittkau et al. 2015b]. Then, among other things, the HTC Vive², a head-mounted display (HMD) suitable for the mass market, was launched 2016. Soon after its release, the HTC Vive was supported by ExplorViz [Häsemeyer 2017]. Building on the existing technologies and hardware, our goal is to extend the existing VR experience of ExplorViz with the ability to use it with multiple users at the same time.

This thesis is the result of a Bachelor project. Daniel König is participating in this project, too. König for his part had the same goal to expand ExplorViz with collaborative VR, but was working with the Oculus Rift³ [König 2018].

1.1 Motivation

Even with the help of visualization, exploring unknown software is a potentially challenging and time-consuming task. In order to train employees or in the hope that groups of two will achieve better results in the field of software development, pair programming is often used today. We want to take up the idea of pair programming and use it for exploring software landscapes in VR with ExplorViz. We further develop the already existing VR extension of ExplorViz to add the possibility to explore software landscapes with several users at the same time. Our goal is to provide the users with many natural and intuitive ways to

¹<https://www.explorviz.net/>

²<https://www.vive.com/de/>

³<https://www.oculus.com/rift/>

1. Introduction

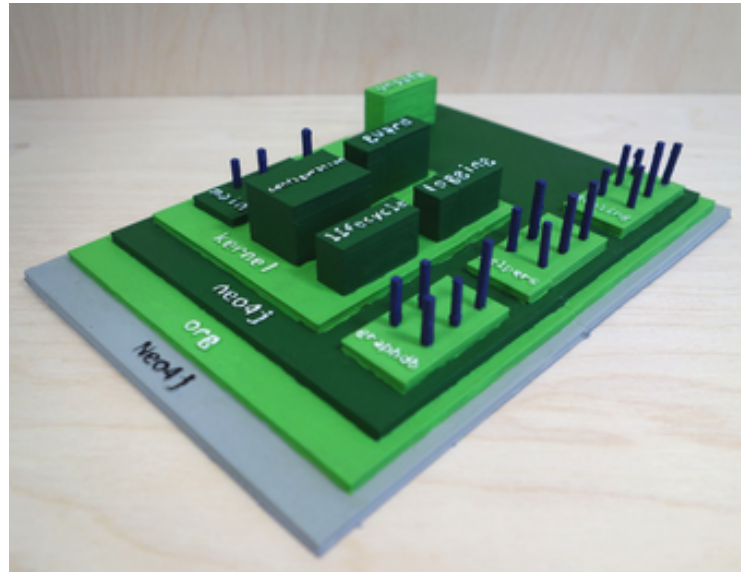


Figure 1.1. A model of a software (Neo4j) exported from ExplorViz and created in a 3D printer

navigate and explore software with ExplorViz. For this we use the HTC Vive including controllers as cutting-edge technology.

An analogous concept has already been tested by printing software models from ExplorViz [Fittkau et al. 2015a], represented in Figure 1.1. Here, users can physically interact with the model and experience a natural way of viewing software models. Such a physical model is particularly suitable for the use in small groups. Our implementation in the VR extension of ExplorViz offers the same advantages, but also adds the possibility to manipulate the model, e.g. to open or mark closed software packages in an application, and to view contents in detail.

In addition to exploring software for new employees, this technology also provides the ability to experienced software developers to visualize data streams and problems in the software. Such problems could be dead code or a bad ratio between cohesion and coupling.

1.2 Goals

Timm Häsemeyer has done significant work on the current state of ExplorViz’s VR mode, particularly with regard to the support of HTC Vive [Häsemeyer 2017]. This thesis follows up on Häsemeyer’s thesis and deals with the extension of ExplorViz. Our main goal is to allow several users to use VR to explore software landscapes collaboratively in ExplorViz.

G1: Concept

Before we start with the implementation it is important to know how the VR extension should technically be realized for multiple users. On the one hand, approaches for the architecture need to be developed, and on the other hand, a concept about whether and how users can interact with the software and other users. Additionally we need to develop a concept for a good usability and come to a decision concerning general design aspects.

G2: Implementation

As soon as we know how the software should be structured conceptually, we have to find out the necessary changes and additions which need to be done in ExplorViz. The compulsory part of the implementation is that several users can use ExplorViz in VR mode at the same time. This means especially that there is a synchronized model, which can be manipulated by any user.

For this goal we are building on top of an existing frontend extension and develop a new backend extension.

G3: Evaluation

If the features developed under G1 have been sufficiently implemented, it is important to evaluate them in order to assess the practical benefits of this approach. A special focus in the evaluation will be placed on user-friendliness. In addition we are trying to figure out whether collaborative VR is suitable for working with ExplorViz.

1.3 Document Structure

In Chapter 2 follows an introduction to the relevant foundations and technologies for this work. In Chapter 3 we present our concept which is then used as a basis for Chapter 4 in which we explain details of the implementation. In Chapter 5 we explain the results of a conducted usability experiment. After we take a look at related work in Chapter 6 we finish this thesis with a conclusion and possible future work in Chapter 7.

Foundations and Technologies

In the following, the fundamental technologies of this work are introduced.

2.1 Virtual Reality

The use of VR is fundamental to this work. VR tries to let a computer generated world feel as real as possible for a user [Billinghurst et al. 2001]. Already in 1968 a head-mounted display (HMD), known as 'Sword of Damocles', was developed [Sutherland 1968]. Based on this, there have been numerous technical developments in the realm of virtual reality.

The goal of virtual reality is to be as immersive as possible [Robertson et al. 1997]. Virtual reality tries to decouple the user from the real world as much as possible. On the other hand, it is the goal of augmented reality to complement the real world with virtual components, which in general are a lower immersion offers.

2.2 HTC Vive

We use the HTC Vive VR system shown in Figure 2.1 as hardware for generating virtual reality. It consists of a headset called HTC Vive, two controllers and two base stations. The base stations, which are ideally located in opposite corners of the room, emit laser beams that are invisible to the human eye. The controllers use those laser beams to determine their own position. The Vive itself has two screens behind lenses, each with a resolution of 1080 x 1200 pixels. One screen provides the images for one eye and by slight differentiation in the images of the screens the impression of spatial vision [Ohzawa 1998] arises. The base stations are emitting infrared light. This is used to determine the position of the headset which has many built-in tracking sensors. The Vive also has an integrated camera, microphone and audio connector.

The two controllers of the Vive VR system are identical so that both controllers can be held with either hand. The position of the controllers in the room is also determined with the aid of base stations and numerous tracking sensors. For additional interaction possibilities, the controller has classic keys, a trigger and a pressable trackpad.

¹<https://www.electronicproducts.com/uploadedImages/Multimedia/Video/HTC-Vive.JPG>

2. Foundations and Technologies



Figure 2.1. Vive VR system with the Vive(center), the controllers (bottom left/right) and the base stations (top left/right)¹

2.3 ExplorViz

ExplorViz is a software for the visualization and monitoring of software landscapes [Fittkau et al. 2017] that has been under development since 2012. In 2017 a structural change from a monolithic architecture to an architecture organized in microservices [Zirkelbach et al. 2018] was realized in ExplorViz. The most important components here are the backend, which is mainly written in Java² and is classically installed on a server and the frontend, which uses the Ember.js³ web development framework. The Javascript⁴ code of the frontend is executed by the user's browser. Through this architecture the development of extensions (e.g. for VR) is very flexible and through the use of Javascript it is possible that ExplorViz can be used system-independently with a modern browser.

For visualization, ExplorViz uses two different views [Fittkau et al. 2015c]. In Figure 2.2 the landscape view of an example software is depicted. The landscape view is a two-dimensional view of a software landscape and is particularly suitable to get an overview of the software landscape. There are systems (gray), nodes (green) and applications (blue). The communication between software is represented by orange lines, where the thickness of the lines correlates with the number of calls it represents. Double-clicking with the left mouse button on a software leads to the application view.

For this work, the application view shown in Figure 2.3 is of greater interest. The application view represents a three-dimensional model of the software and offers many

²<https://www.java.com/de/>

³<https://www.emberjs.com>

⁴<https://www.javascript.com/>

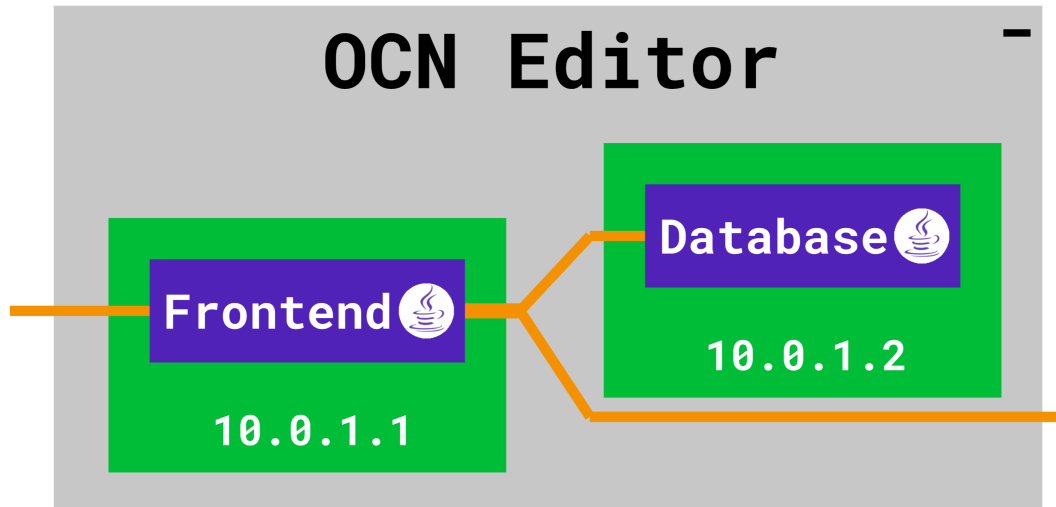


Figure 2.2. Opened system of the landscape view in ExplorViz

interaction possibilities. Software packages are shown in green, which in turn can contain packages or individual classes (blue). The height of the blue blocks indicates the number of objects belonging to the class. Here, too, the communication between objects is represented with orange lines. One can highlight individual classes and components in red or call up additional information for a component or class. This kind of representation is supposed to be a metaphor for a three-dimensional city [Caserta et al. 2011], where the classes here represent houses and communication between classes is comparable to streets.

2.4 ExplorViz VR extension

In addition to the core software of ExplorViz, we are building on top of the VR extension. Timm Häsemeyer developed this frontend extension to enable the use of the core features of ExplorViz with the HTC Vive [Häsemeyer 2017]. For our work we are building on top of the extension in version 1.1⁵.

In version 1.1 the user enters a virtual world, which consists of a square floor. On top of the floor a 3D adaption of the landscape model is placed. Everything apart from the floor and the landscape is white. Interaction with objects in the world is done by using the HTC Vive controllers (see Figure 2.4).

Therefore, a 3D model of the controllers with additional rays is shown in the virtual

⁵<https://github.com/ExplorViz/explorviz-frontend-extension-vr/releases/tag/v1.1>

⁶<https://docs.unity3d.com/Manual/OpenVRControllers.html>

2. Foundations and Technologies

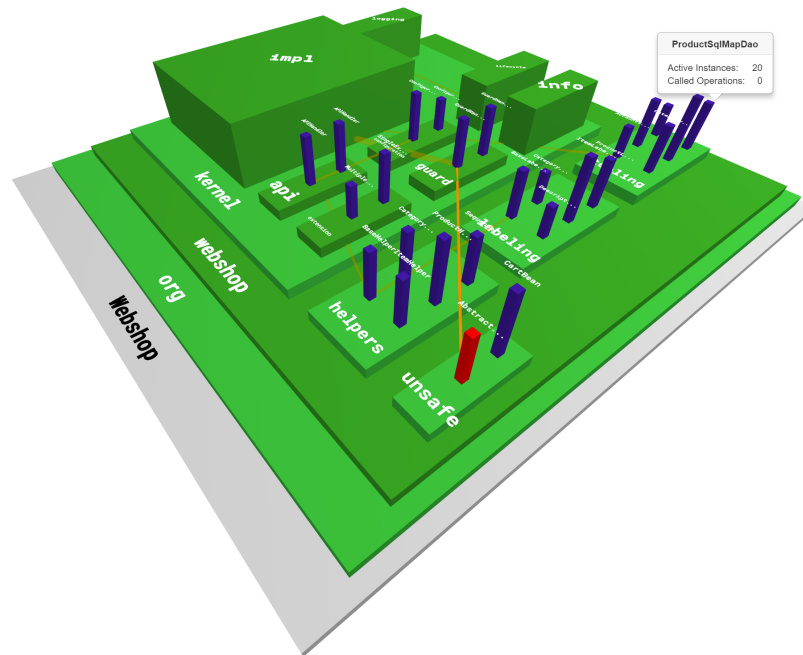


Figure 2.3. Application View in ExplorViz

world. The position of the controllers is set relative to position of the HMD and thus corresponds to the position of the controllers in the real world. Even though the two HTC Vive controllers are physically identical, the controls differ for the right and left controller.

As part of a developed room concept, a user can move through the virtual world by moving in the real world. Since the physical space the users are located in is limited, users can additionally move in the virtual world by teleporting. Therefore a user points his left controller on the floor and actuates the trigger (see ⑦) of the left controller. As a visual feedback, a blue circle is shown on the floor at the position where a user can teleport.

Elements of the landscape, such as systems, can be opened or closed by pointing the right controller on those elements and pressing the trigger on the right controller. If a user tries to open a non-empty application, a 3D model, which is identical to the 3D model of the frontend (see Figure 2.3), is shown. An application can be moved in the virtual room by pointing the right controller on the application and clicking and holding the trackpad (see ②) on the right controller. This will bind the movement of the selected application to the movement of the right controller.

Components of an application can be opened and closed just like elements of the landscape. Classes and closed components can be selected by pointing the left controller

2.4. ExplorViz VR extension

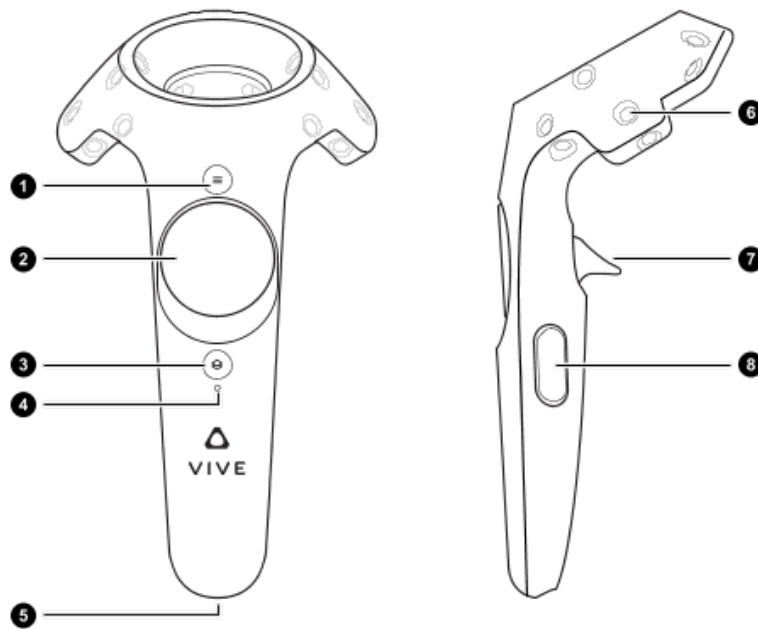


Figure 2.4. Elements of HTC Vive controller⁶

on them and actuating the trigger on the left controller. When pointing the right controller on a class or component and clicking the trackpad, additional information to that element is shown as a little note next to the right controller.

Concept

Our goal is to enable multiple users to work simultaneously in ExplorViz to explore and experience software using VR software. Therefore, we develop a concept for the overall architecture and from there on explain necessary changes in the backend and frontend.

3.1 Architecture

ExplorViz uses a microservice architecture [Zirkelbach et al. 2018] and thus is partitioned in several components. The core components are the backend and frontend (see Figure 3.1). The frontend receives resources from the backend. Therefore, a REST API with underlying HTTP is used. To add additional features to ExplorViz extensions can be developed, which will be covered in the following sections.

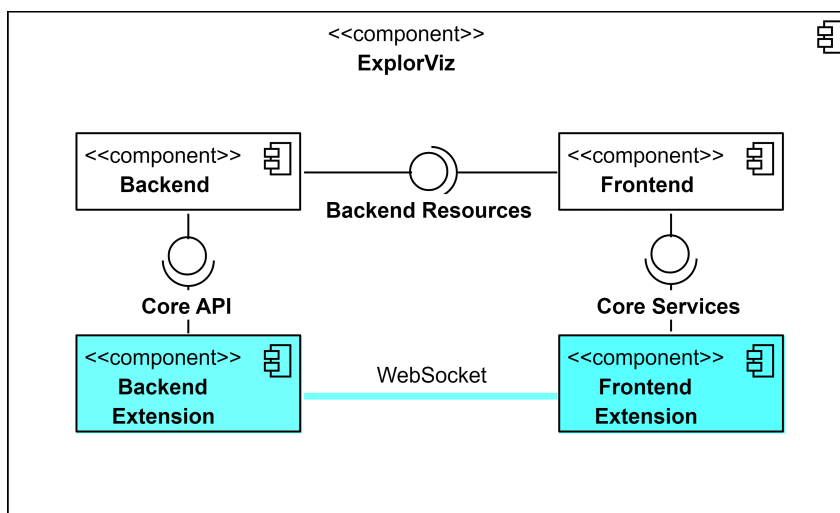


Figure 3.1. Core ExplorViz components and envisioned extensions (blue)

Usually resources are transferred between a backend extension and a frontend extension with REST, just like between the backend and frontend. However, since only very small

3. Concept

amounts of data are likely to be transferred for each time data is sent, we want to use a protocol with a smaller overhead. Such a protocol would be UDP. However, there are currently no practical technologies available to use UDP for web applications due to security reasons. Therefore, we strive to use TCP.

Since we want to use a programming language independent protocol, which allows bidirectional communication, we are using WebSocket. The WebSocket protocol provides a full-duplex communication. Still, only a single TCP connection is needed for the connection between one client and the backend extension. For this reason using WebSocket is a good choice for real-time web applications [Liu and Sun 2012].

To be able to use ExplorViz with several users at the same time, it is important that information is available among the participants of a session and can be shared. In particular, this includes the position data including the orientation of the software models and user models.

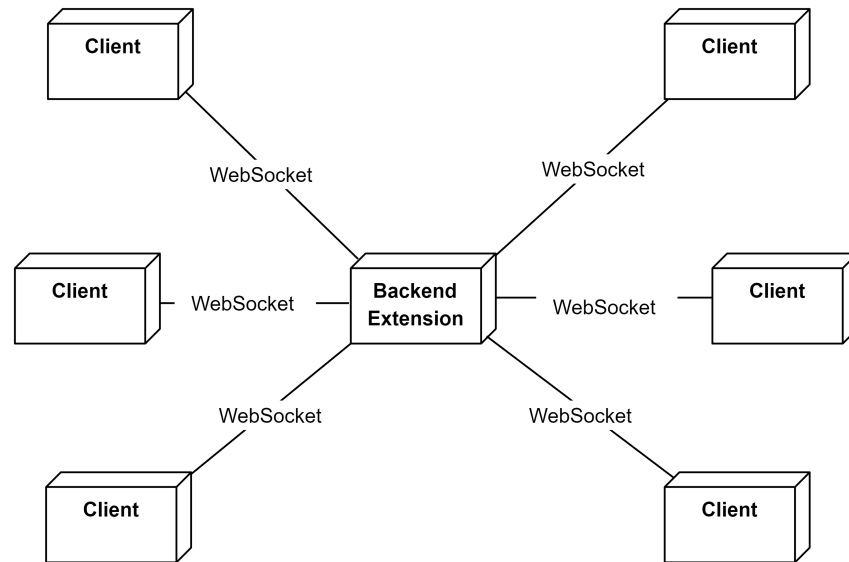


Figure 3.2. Envisioned star topology

The current architecture of ExplorViz [Zirkelbach et al. 2018] suggests a star topology [Roberts and Wessler 1970] for the required network. Here, the backend extension is the central node, which provides all necessary position data. If the state of the model changes due to the interaction of a user, the updated information can be sent to all other users.

3.2 Backend extension

In this section we present our approaches for the development of the backend extension.

Extension Development

The backend extension can be build with Java and *Gradle*¹. The extension can access the core functionalities of the backend. Since there does not a exist a backend extension for VR at this point, we need to develop a new one. For this purpose, an existing dummy extension with basic functionalities can be used.

Synchronization

The backend extension is the central node of the envisioned star topology. Therefore, its most important task is to send updates of one connected client to all other connected clients.

From there on it is important that the backend extension keeps track of the current state all the time. This information is for example needed when a new user connects. In order to synchronize the views all relevant information need first to be stored by the backend extension and then sent to the new user.

3.3 Frontend extension

In this section we present our approaches for the development of the frontend extension. Since there does already exist a frontend extension with version 1.1 which we will use as a basis, the following changes will describe an envisioned version 2.0.

Extension Development

The frontend uses *Ember.js* and *npm*² as a package manager. The frontend extension can be developed separately from the frontend. The frontend and the frontend extension are merged when the software is compiled. Therefore, *npm* is used to link the frontend and the respective extension.

We do not need to create a completely new extension, since we will build on top of the extension in version 1.1, which was developed by Häsemeyer. The features of this version are explained in Section 2.4.

General concept

The benefit of collaborative VR in ExplorViz should be granted by the fact that multiple users instead of only one can interact with and manipulate models of software. In Figure 3.3 it is shown how a collaborative use of ExplorViz can look like. We also consider that users can spectate, but mostly focus on features for users who want to interact with the models themselves.

In the context of user-friendliness, we attach great importance to the fact that manipulation of the model by a user will be carried out as quickly as possible. This in return should

¹<https://gradle.org/>

²<https://www.npmjs.com/>

3. Concept

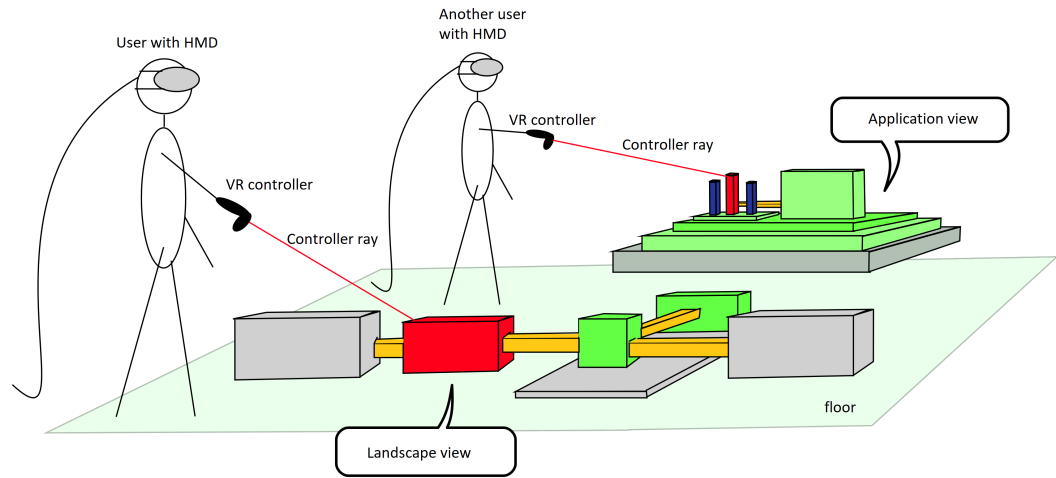


Figure 3.3. Possible visual concept for the collaborative use of ExplorViz [Häsemeyer 2017]

in cooperation with the backend extension lead to an update of the view for all other users. In particular, when a user for example moves the model, the positional update for other users should not only take place after the move has been completed, but also during the move, in order to obtain the cognitive map [Misue et al. 1995] for all users.

In addition to these essential functions, communication options are also desirable. One such option is to highlight elements of an application for other users. We assume that most users want to talk to each other while using our technology. However, there already exists a lot of software for voice communication which can be used in addition to ExplorViz.

In order to facilitate natural communication, we depict users in virtual space using 3D models. Accordingly, the positions of these models must also be updated as required.

Menu

In addition to the mentioned core functionalities it is important to us to implement additional functionalities to improve the overall usability. Since the amount of input buttons is very limited, we plan to add a menu (see Figure 3.4). The menu should be hierarchically structured. When opened with a menu button, it should first show an overview of all existing submenus, which provide different functionalities. Navigating through the menu can be realized by pointing the controller on a textfield and actuating the trigger to select an entry. In our opinion changing the height of the camera and controllers, moving the landscape, spectating and changing between working alone or with others are important functions.

In the camera menu the user should be able to click on buttons to move his camera including his controllers up or down. In the landscape menu the user should be able to

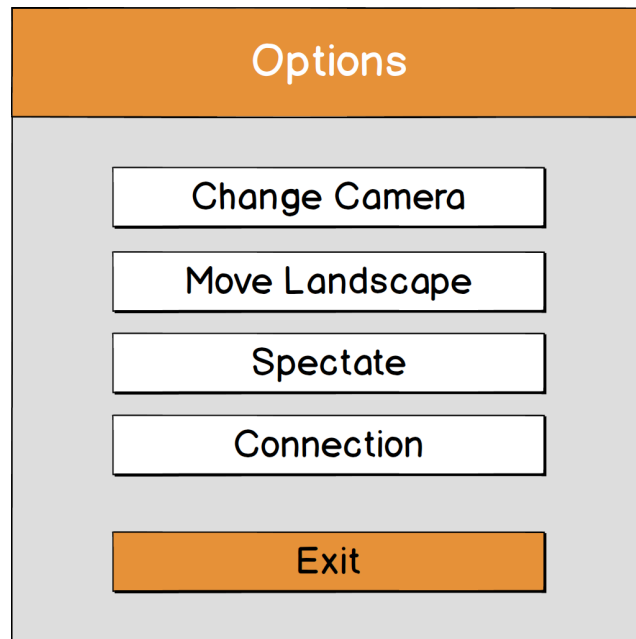


Figure 3.4. Draft of overview menu

move the landscape along the x-, y- and z-axis and rotate the landscape. The spectate menu should allow the user to set his own camera position to the camera position of another user. The connection menu should show the current connection status and allow the user to switch between working alone and working collaboratively.

Text messages

Version 1.1 of the frontend extension does not include any text messages for the user. However, these could improve the usability. The web interface of the frontend shows little boxes with messages for example when an application does not contain any data. We would like to adopt the already existing messages of the frontend and add our own messages which are specific for the collaborative environment. The important messages, which we mostly adopt from the frontend such as missing data for an application, should be displayed directly in front of a user for a few seconds. This would make sure that important messages are not missed. Less urgent messages such as the information that a user disconnected or is spectating should be displayed in the top and thus not influence the usability.

A visual concept for such messages is shown in Figure 3.5.

3. Concept

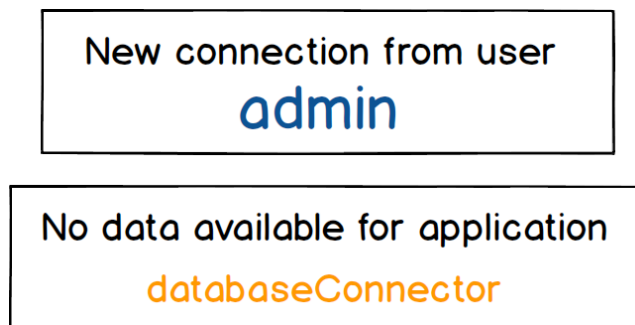


Figure 3.5. Draft of messages which provide a user with additional information

Implementation

This chapter focuses on the implementation of the backend and frontend extension. In the following adaptations and additions to the implementations are explained and reasoned. Where applicable, pictures and small code snippets are provided to help illustrate the current state of the implementation.

4.1 Backend Implementation

This section describes how the backend implementation is extended. As a starting point a dummy extension is used, which provides sample code, e.g. for simple resource exchange. Most of this sample code is removed until only the code for linking with the backend remains. From there on WebSocket technologies for communication with the frontend extension are implemented. Then models are implemented to store the current state of users, landscape and applications. These are then used for example to allow new users to connect and have the same synchronized environment like all already connected users.

4.1.1 Connection to Frontend Extension

Our main class, which contains the methods for communication with the frontend extension, extends the class `WebSocketServer`. This class is provided by a java library and allows us to use WebSocket functionalities. The WebSocket is started when the backend extension is loaded. Additionally, a new thread is started. This thread is running until the WebSocket is closed again and checks constantly if new messages need to be sent to a user. To limit the amount of traffic, messages are first collected in arrays before they are sent. Then, maximally 90 messages containing those arrays are sent out per second to a client. Every message uses the JSON format and contains a key entry called `event`. This entry specifies what kind of information a message holds.

Since the same format is used for the received messages from the frontend, a switch statement is used to handle different types of messages. Most events, like the opening of an application, are forwarded to all other users. Additionally, the contained information about model updates are saved and can be used to later on send a new user the current state of the virtual world.

4. Implementation

4.1.2 Synchronization

As mentioned in the previous section, the important data of incoming messages usually needs to be stored. When a new user connects, a model is created which contains the name, connection state, a color, information about the used controllers and highlighted entities. Lastly a list which maps the users to the corresponding WebSocket connection is updated.

Information about the structure of the landscape or applications can be requested from the backend. The state of landscape systems and nodes is stored in Hashmaps which map the corresponding identifier of the system or node to a boolean value, which tells whether a system or node is opened. Additionally, positions and quaternions of the landscape and all open applications are stored. For applications the current state is saved analogously. Since components of classes can be highlighted, the information of which user highlighted which elements is stored additionally.

4.2 Frontend Implementation

This section focuses on the main changes which were made to the implementation of the frontend extension for VR. The implementation is using the version 1.1 as a starting point.

4.2.1 Connection to Backend Extension

As mentioned in the concept we use a WebSocket connection to interchange data between the frontend extension and the backend extension. Therefore, we use the package *ember-websockets*¹ which allows us to use WebSocket functionalities. Our first approach was to establish a WebSocket connection the backend extension when a user enters the 'VR' tab in ExplorViz. While there were no technical problems in doing so, we chose to let the user decide if and when he wants to work together with other users.

Listing 4.1. Preparation of WebSocket connection to backend extension

```
1 connect() {  
2   this.set('state', 'connecting');  
3   ConnectMenu.setState.call(this, 'connecting');  
4   this.set('updateQueue', []);  
5   this.initSocket(this.get('host'), this.get('port'));  
6 }
```

The user can change his connection status via the connection menu where he can also see his own connection status. When a user actuates the connect button in that menu, the connect function (see Listing 4.1) is called. Thus the connection state of the user changes

¹<https://github.com/thoov/ember-websockets>

4.2. Frontend Implementation

and therefore the message in the menu is updated. Lastly, as it can be seen in Listing 4.2, the WebSocket connection is initialized.

To establish a WebSocket connection to the backend extension, an IP address including a port is needed. We chose to outsource this information into a configuration file using JSON as a format (see line 1-4).

Listing 4.2. WebSocket Initialization

```
1  Ember.$.getJSON("config/config_multiuser.json").then(json => {
2      host = json.host;
3      port = json.port;
4  });
5
6  ...
7
8  initSocket(host, port) {
9      const socket = this.get('websockets').socketFor('ws://${host}:${port}/');
10     socket.on('open', this.openHandler, this);
11     socket.on('message', this.messageHandler, this);
12     socket.on('close', this.closeHandler, this);
13     this.set('socketRef', socket);
14 }
```

After the socket is initialized (line 9) the respective handlers are initialized (line 10-12) to allow for a message exchange with the backend extension. If the WebSocket connection can be established, messages can be easily sent to the backend extension with help of the variable named `socketRef` (see line 13).

4.2.2 Main Loop

With the main loop we refer to a function (see Listing 4.3) which is called every time an image is rendered. Since the display refresh rate of the HTC Vive is 90 frames per second, the main loop is called up to 90 times per second. In the main loop everything is contained that needs to be updated or checked for updates regularly. Here, the function for changing the camera position to another user's camera position is called if the user is currently spectating. Additionally, the updates for the orientation of the displayed usernames are originating here.

4. Implementation

Listing 4.3. Main Loop

```
1   if(this.get('userID') && this.get('state') === 'spectating') {
2       this.spectateUser(); // follow view of spectated user
3   }
4
5   this.updateControllers();
6
7   if(this.get('userID') && this.get('state') === 'connected' || this.get('state'
8       ) === 'spectating') {
9       this.updateUserNameTags();
10      this.update();
11  }
12  this.render2();
13
14  //send messages like connecting request, position updates etc.
15  if(this.get('state') !== 'offline')
16      this.sendUpdates();
17
18  this.set('lastUpdateTime', this.get('currentTime'));
```

4.2.3 Landscape

The landscape provides an overview of software landscapes and is a 3D adaption (depth was added) of the 2D representation, which is used for the ExplorViz frontend. In the implementation we distinguish between the pure `vrLandscape` and the corresponding `vrCommunications`, so systems and communications are handled separately. Those two components are added in a group named `vrEnvironment`. However, in the following we will refer to the `vrEnvironment` as landscape, which therefore also includes the communication.

Landscape Movement

In version 1.1 the landscape is only movable by mouse and keyboard. Also, when the landscape is manipulated, e.g. a system is opened, the landscape is always re-centered to the middle of the floor. We chose to keep the original position of the landscape as the default but wanted to incorporate the option to move the landscape via the controllers. Therefore, the menu shown in Figure 4.1 was added. It is possible to interact with the yellow menu elements by pointing the right controller at them and clicking the trigger on the right controller. The yellow arrows on the top left allow changes in height. The four central yellow arrows allow the movement along the x- and z-axis. With the curved

4.2. Frontend Implementation

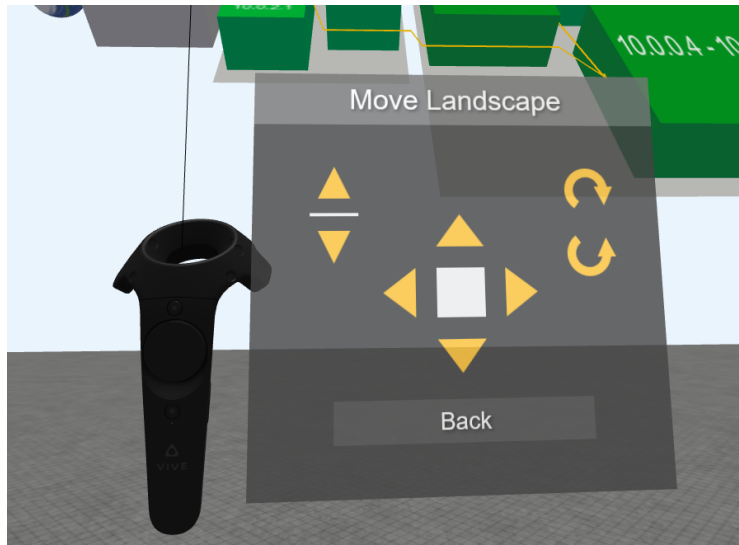


Figure 4.1. Landscape menu

arrows on the top left it is possible to rotate the landscape towards or away from the user (provided the user is standing in front of the landscape).

Listing 4.4. Centering of the landscape

```
1   vrEnvironment.position.x += centerFloor.x - centerLandscape.x + this.get('
    environmentOffset.x');
2   vrEnvironment.position.z += centerFloor.z - centerLandscape.z + this.get('
    environmentOffset.z');
```

Since the size and expansion of the landscape can change whenever systems or nodes are opened or closed, it is necessary to center the landscape around the desired position. In Listing 4.4 the most important part of recentering is shown. An offset, which is set to 0 when ExplorViz is started, contains information about the desired landscape position in relation to the center of the floor. The offset is changed whenever the landscape is moved. To center the landscape around the desired position, we calculate the center of the floor, the center of the landscape and add the existing offset eventually.

Landscape Interaction

Whenever a user manipulates the landscape, an update message needs to be sent to the backend extension containing the new state of the landscape model. To achieve this, an event is triggered when for example a system is clicked and thus its status changed. The event contains the information whether a system, nodegroup or node was opened or closed.

4. Implementation

The event is caught and the important data is put into a JSON message which is then added to an array. All messages of this array are eventually sent out together to the backend extension.

4.2.4 Applications

The applications that are displayed in the application view provide important information to the user and offer a lot of interaction possibilities in virtual reality. To make good use of the application view for collaborative VR, we first allow multiple applications to be opened at the same time and then go over to synchronizing user interactions concerning applications. Lastly, we synchronize and extend the option to highlight components and classes of an application.

Multiple Applications

The frontend is designed to allow one open application at a time. This is sufficient for the existing web interface because a single user can switch between the landscape view and application view to explore an application in detail. For our collaborative solution it is important for the productivity to allow multiple users to work with different applications at the same time.

In version 1.1 the variable `application3D` among others is used to access the visual representation of an application. In order to handle more than one open application at a time we use a `Map` as a data structure. This variable maps numerical application identifiers to the corresponding application object.

Every application consists of a gray foundation as a root component. In version 1.1 this foundation is added and removed by calling existing methods of the frontend.

Listing 4.5. Creation of a foundation

```
1   const foundation = this.get('foundationBuilder').createFoundation(  
    application, this.get('store'));  
2   this.get('foundations').set(application.id, foundation);
```

We managed to reuse the foundation-builder of the frontend to create a foundation for a new application as shown in Listing 4.5, too. However, we need to keep record of all existing foundations to be able to remove those later on if necessary with help of an own function. For better reuse of existing code we advise an update of the foundation-builder, which is part of the frontend, so that it can handle multiple applications at a time.

Application Movement

Closing or opening a component or class of an application is handled very similar to the described mechanism for landscape interactions.

4.2. Frontend Implementation

However, applications can be moved by moving the controller as opposed to the movement of the landscape. First, we considered to send the position of a moved application continuously just like the positions of the controllers are sent. However, this is not necessary because the position of a moved application is bound to the position of a controller.

As shown in Listing 4.6 the only required information is which application (line 4) is bound to which controller (line 9) and additionally the initial positions of the application and controller (lines 5-8). The information which user is moving the application is added by the backend extension.

Listing 4.6. Sent message when an application is grabbed by a user

```
1 let appMsg = {  
2   "event": "receive_app_bound",  
3   "appID": appID,  
4   "appPosition" : appPosition.toArray(),  
5   "appQuaternion" : appQuaternion.toArray(),  
6   "controllerPosition" : controllerPosition.toArray(),  
7   "controllerQuaternion" : controllerQuaternion.toArray(),  
8   "isBoundToController1" : isBoundToController1  
9 }
```

Highlighting

We want to keep the functionality to highlight classes and closed components. For the collaborative use it is important to us that every user can see what other users have highlighted to simplify communication. In version 1.1 objects are always highlighted in red. Naturally, we choose to highlight elements in the color which is assigned to a user.

When a user highlights an object, a message containing the identifier for that object is sent. Nested components in applications have alternating colors which cannot be requested from the frontend easily. Therefore, we send the color of the unhighlighted component along with the highlighting message to ensure that the original color can be reassigned when the component is not highlighted anymore.

4.2.5 User Representation

To allow for natural interaction between users it is crucial to implement a visual representation of a user for other users. We chose to only visualize the head respectively the HMD which another user is wearing and the two controllers for each user. Additionally, we added usernames on top of other users to allow easy identification for the use with multiple users.

To keep track of all other users, a user object is created every time a new user connects. This object contains data including an identifier, name, connection state, color and the 3D

4. Implementation

Models for HMD and controllers.

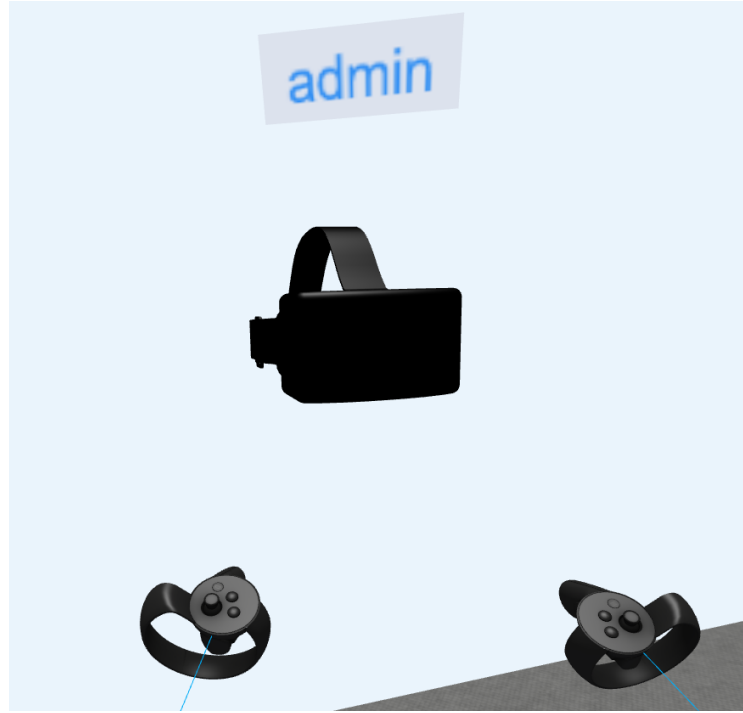


Figure 4.2. Visualization of other user including HMD, controllers and & colored name

HMD & Controllers

In version 1.1 the visualization of the user's own HTC Vive controllers was already implemented. This implementation uses OpenVR², which is part of SteamVR. Thankfully, OpenVR also contains textures for a generic HMD and the Oculus Rift controllers. On receiving a message from the backend about a new connected user, an object for the camera is created and the corresponding texture is applied. The same is done for the controllers but here it needs to be distinguished between HTC Vive Controllers and the left and right controllers of the Oculus Rift to use a matching texture.

For all visually identical objects (e.g. HMDs) the same loaded texture is applied in contrast to loading a new but identical texture for every object. It should be noted that as a consequence manipulating the mesh of one Object (e.g. changing transparency) leads to a change of all meshes which use the same texture.

²<https://github.com/ValveSoftware/openvr>

4.2. Frontend Implementation

Username

In scenarios where three or more users are working together it is particularly important to add intuitive mechanisms for users to identify each other. As a first step usernames were added on top of the virtual HMD of every other user. This username is matching with the name used for the login into ExplorViz.

To visualize the username a canvas is used which adapts in size with the length of the username. We chose to make the background of this canvas light gray and mostly transparent to improve readability on the one hand and on the other hand avoid the visual blocking of other important models.

It was made an effort to use the canvas as a texture for a sprite to display usernames. Sprites have the advantage that they always face towards the user's camera and should therefore provide constant readability of the usernames. However, this approach was not satisfying. If the user was rotating his HMD to the side, the sprite would perform the same rotation. This lead to unreadable usernames or a visual interference of the other user's HMD and his sprite containing the username.

Instead the canvas is used as the texture for a plane and eventually added to the scene on top of the HMD.

We wanted to display the username always on the same position above another user's HMD. Furthermore it should only follow the user's camera by rotating around the y-axis. To achieve this behavior we call a function of *three.js* in every iteration of the main loop to let the other usernames always face towards the user.

To improve identification of other users even further every user is assigned a color by the backend. This color is used as a text color for the usernames and for highlighting objects of an application. To get a quick overview about all connected users and their colors, an overlay with a userlist can be opened by pressing the grip button on the left controller.

4.2.6 Menu

In addition to a pure implementation of a mode for multiple users it was our goal to enrich the user experience with intuitive ways of interaction. The limited amount of input buttons lead to the development of a menu which is attached to the left controller, displayed in Figure 4.3. This menu can be opened with a click on the left menu button. Initially, an overview with all available submenus is shown. The menu button can also close the menu or jump back to the main menu if one of the submenus was selected.

The right controller is used to interact with the menu. The user can point the right controller and its ray on the menu. When hitting a clickable entry in the menu it is highlighted and signalizes that it can be selected by clicking the trigger on the right controller. In its current state the user can adjust his height, move the landscape, spectate other users or change his connection status. These submenus will be explained in detail on the following pages together with their corresponding functionality.

4. Implementation

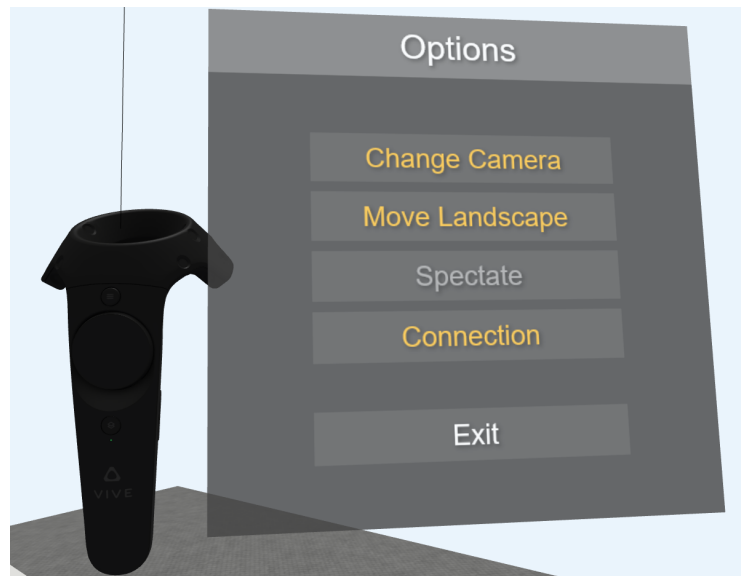


Figure 4.3. Menu attached to left controller

4.2.7 Text messages

As described in the concept, text message should give a user additional information to improve usability. Messages about state changes (e.g. disconnect) of other users are shown at the top edge of the display. We animated the message such that it moves from outside the users field of view to the top edge of the screen. This minimizes possible disturbances for the user. The message is shown for a few seconds before it disappears. Even though this message should usually not block important information, we chose to make this message 30% transparent. With this configuration the text messages still remain readable.

In addition to connection messages we implemented more important messages which we refer to as hints. Among other things hints can tell a user why he can not open an application or that he can not move an application due to another user who grabbed the application first. To be sure a user notices the hints we place them centrally in front of him. However, we do not want to let hints pop up directly. Therefore, we chose to also animate hints and make them 30% transparent. Hints start as a small stripe before they widen and eventually the full text is visible. The hint stays for 2-3 seconds until it disappears.

4.2.8 Teleportation & Height Adjustment

In version 1.1 teleporting through the virtual room is already implemented. This feature is used by pointing the left controller with its black ray on the floor and pulling the trigger on the left controller. As a visual feedback for the user a blue circle is shown on the

4.2. Frontend Implementation

floor at the position where the controller points whenever teleportation is possible. To realize the teleportation in version 1.1 the visualized objects including floor, landscape and applications were shifted relative to the user. Thus, the user never changed his position but got the impression of being teleported.

Since the position of a user should be exchanged with other users, this approach was not practical. Keeping this approach would have lead to the use of offsets, which we wanted to avoid. However, there are restrictions to changing the positions of the camera and controllers. Changing the position of the respective object will have no effect. As a solution the objects for the camera and controllers are added to a group named user. This group, working as a wrapper for the user's hardware models, can be moved by changing its position. This positional update will be sent to the backend with the next iteration of the main loop without the use of any offsets.

Listing 4.7. Implementation of teleportation

```
1  teleportToPosition(position){  
2      const cameraOffset = new THREE.Vector3();  
3  
4      cameraOffset.copy(this.camera.position);  
5      cameraOffset.y = 0;  
6      this.user.position.subVectors(new THREE.Vector3(position.x, this.user.position  
7      .y, position.z), cameraOffset);  
8  }
```

The adjustment of the user's height (including camera and controllers) is realized by moving the group user, too. The height can either be changed via the left controller menu or by using the up and down arrow keys on the keyboard. Additionally, the user group can be moved to the left or to the right with the corresponding arrow keys, which is a useful feature for developing purposes.

4.2.9 Spectating

We focus on features which allow natural interaction with landscape and applications. However, we wanted to add additionally a basic feature to allow users spectating each other. This feature allows a user to select another non-spectating user via the spectating menu to spectate. Enabling spectating will lead to a small notification for all other users. The spectating users are no longer visible but are listed in the user list under spectating.

The spectating user's camera position is continuously set to the camera position of the spectated user. To allow for a certain degree of free movement, the spectating user can still look around on his own by rotating his head. We still expect that for some users spectating could be uncomfortable. We chose to disable the spectating again whenever the user closes the menu to avoid situations where the user does not know how to quit spectating.

4. Implementation

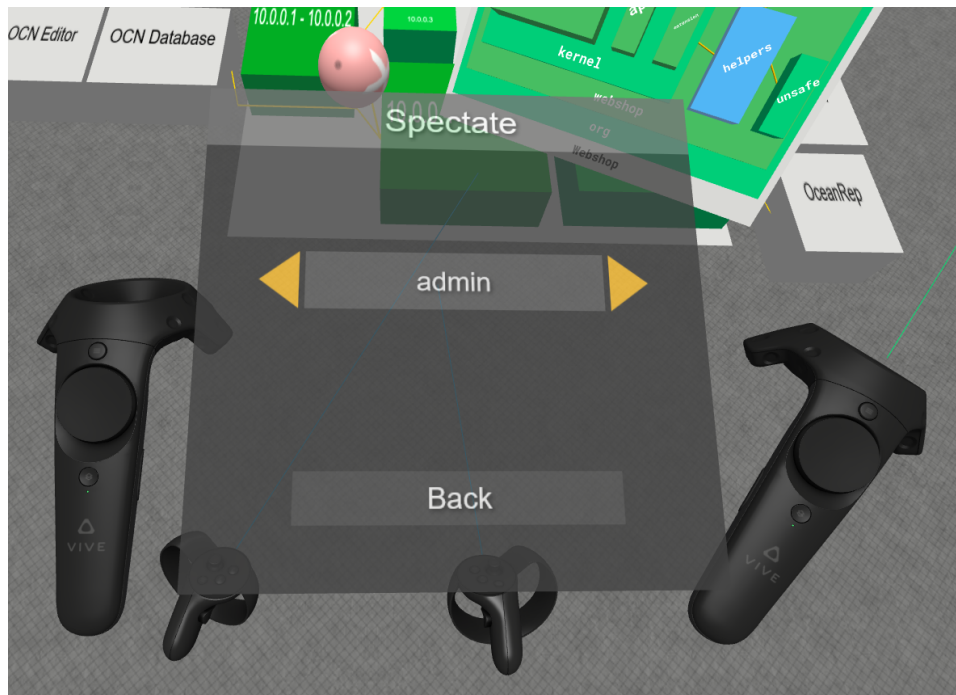


Figure 4.4. View of a spectating user

The spectating user can estimate where the spectated user is looking because the controllers of the spectated user are visible for him (see fig Figure 4.4). For a spectated user it is not possible to interact with the landscape or an application. This is done because a spectating user is not visible and thus allowing interaction could lead to confusion for other users.

4.2.10 Further Adjustments

The texture for the floor was originally an upscaled picture of a gray floor. As a result the floor looked pixelated. Therefore, we changed the texture to an also gray but much sharper and clearer texture. Additionally, we figured out that the floor space was very small for multiple users. To adapt for this scenario we enlarged the floor area.

Evaluation

In this chapter we describe our procedure for the usability evaluation. We explain our goals, the used questionnaire, the experimental setup and the execution of the experiment. After we present the results of the evaluation, we go over in a section for discussion and point out possible threats to validity.

5.1 Goals

This usability experiment should give some insights on how well ExplorViz can be used collaboratively with VR. We decided against an experiment which compares our approach with another one, e.g. working together in front of a monitor because for this it would be hard to achieve comparable conditions. Additionally, collaborative VR for software visualization is a new field of research and thus first can be evaluated in isolation to receive general feedback on this technology.

The conducting probands will execute tasks collaboratively in groups of two. These tasks are designed to be simple and should predominantly encourage the probands to try the implemented features and get an impression of how working with collaborative VR looks like. Probands are then asked to rate certain aspects of their user experience. These aspects can be classified as follows:

- ▷ General Usability
- ▷ Interaction
- ▷ Visual appearance
- ▷ Spectating

In terms of general usability we are interested in whether our implementation for collaborative VR feels natural to use and if it is suited for team work. Concerning interaction we are interested if the movement through the virtual room and interactions with objects are well implemented. Since we added a visual representation for other users, a menu and text messages we are especially interested if those are perceived as visually pleasing. Lastly, we want to research if spectating is a well received feature.

5. Evaluation

5.2 Questionnaire

The main tool to receive feedback of the probands is a questionnaire. In addition to questions the paper based questionnaire contains information for the probands and tasks which are read out loud. The specific texts and tasks are displayed in Appendix 7.2. In the following we present the main questions and statements of the questionnaire which we use to analyse the results.

Personal Information

After an introducing text the questionnaire contains a page requesting personal information of the probands. These information can be used to correlate given feedback with certain aspects or skills of the probands. The asked questions and statements about personal information are listed in Table 5.1. In addition to those questions and statements probands are asked to fill in their age, profession, subject of study and gender. A3 to A9 are statements or questions which probands can rate subjectively from 0 (not agreeing) over + (slightly agree) and ++ (agree) to +++ (strongly agree).

Table 5.1. Questions and statements regarding personal information

ID	Question / Statement
A1	Do you wear glasses?
A2.1	Do you have any visual impairment?
A2.2	If so, which:
A3	Experience with objectoriented programming
A4	Experience with ExplorViz
A5	Experience with VR
A6	Are you claustrophobic?
A7	Are you afraid of heights?
A8	Do you suffer from seasickness?
A9	How well do you know the other proband?

Rating

At the end of the experiment the probands are asked to rate certain aspects of their user experience. Therefore, they are given two pages of statements for which the probands can tick checkboxes to express their agreement or disagreement respectively. The possible checkboxes for a statement are explained in Table 5.2. It can be seen that there is no neutral

5.2. Questionnaire

choice for the probands. We choose to exclude a neutral option because we want to request tendencies for our new approach. Including a neutral option for an experiment with a small number of probands could lead to inconclusive results. In addition to purely ticking checkboxes probands were free to write down additional comments on a provided line under each statement.

Table 5.2. Checkboxes and their respective meaning for rating statements

Checkbox	Meaning
- -	Strongly disagreeing with the statement
-	Disagreeing with the statement
+	Agreeing with the statement
++	Strongly agreeing with the statement

Statements

In the following we present the statements which probands are asked to rate. We classified the statements in topics in coordination with our goals. The following tables provide an identifier (ID) and the respective statement for each topic (see Table 5.3, Table 5.4, Table 5.5, Table 5.6).

Table 5.3. Statements on interaction

ID	Statement
B1.1	Moving and rotating the landscape is well realized
B1.2	The moving and rotating of the applications is well realized
B1.3	The movement (incl. teleportation) in the virtual space is intuitive
B1.4	Navigation through the menu is intuitive
B1.5	Highlighting objects is a useful feature

5. Evaluation

Table 5.4. General statements

ID	Statement
B2.1	I had the impression that I was in the same room with the other user (positions and state of users, landscape & applications were synchronized)
B2.2	ExplorViz with VR extension is suitable for team work
B2.3	During the experiment (except spectator mode) I felt nausea or something alike
B2.4	I would use ExplorViz with the VR extension again

Table 5.5. Statements on visual appearance

ID	Statement
B3.1	The structure of the menu is intuitive
B3.2	The visualization of other users is well done
B3.3	The movements of the other user were displayed to me without delay
B3.4	The number of text insertions was reasonable
B3.5	Text overlays were clearly readable
B3.6	Text overlays were visually appealing (duration, length, position, color, size, animation)

Table 5.6. Statements on spectating

ID	Statement
B4.1	While spectating I felt nausea or something of the like
B4.2	I would use the spectator mode again

Statements

In addition to the statements probands have the opportunity to write down further remarks and proposals on the last page of the questionnaire. The respective text fields are listed in Table 5.7.

Table 5.7. Further comments text fields

ID	Text field
C1	Proposals for future features
C2	Improvement proposals
C3	Further remarks

5.3 Experimental Setup

In the following we list the used hardware and software configurations. Since the experiment is conducted in an experimental laboratory, the hardware and software was kept identical throughout the experiment.

System Configuration We use three different computers. Two computers run the frontend with installed VR extension. The other one runs the backend with the installed VR extension. In our opinion this represents a distribution of the software close to a setup which could be used in practice.

The relevant specifications of the computers are listed in Table 5.8. The hardware is sufficient to avoid performance issues. In preliminary tests CPU had no performance problems on either machine and the computers which run a frontend produced steady 90 frames per second. Nevertheless, it should be noted that the computer which is driving the HTC Vive is attached to two displays, one monitor and one projector. This can potentially influence frame rates negatively but this setup is providing us with the possibility to monitor easily what the probands are seeing and doing in the virtual environment.

Table 5.8. Hardware configuration which is used for the experiment

	Computer 1	Computer 2	Computer 3
OS	Windows 10 Pro	Windows 10 Pro	Windows 10 Pro
CPU	Intel Core i5-6500	Intel Core i5-6500	Intel Core i5-6500
RAM	8GB	8GB	8GB
Graphics Card	GeForce GTX 1070	GeForce GTX 1070	GeForce GTX 950
Used for	HTC Vive	Oculus Rift	Backend
Additional Notes	Mirrored Displays	3 sensors connected	

5. Evaluation

Room Setup All probands participate in this evaluation in groups of two to test the implemented features. All the used hardware is located in one room and therefore the areas where probands can move are located close to each other but do not overlap.

The proband who uses the HTC Vive has an area to move freely of 3.90m x 3.70m due to space limitations. This area can also hardly be increased because the HTC Vive is connected with a cable to the computer which is positioned in one corner. The base stations are wall mounted in opposing corners of the area for moving at a height of 2.30m above the floor.

The proband who uses the Oculus Rift has an area to move freely of 2.10m x 2m. In comparison to the setup for the HTC Vive this is significantly smaller but again room limitations and short cables for the sensors do not allow for a bigger area. The Oculus Rift is usually shipped with two sensors. However, two sensors are not sufficient for 360 degree tracking. To avoid tracking issues when a proband turns around we use a third sensor. The three sensors are placed on tables in a height of 72cm in the shape of an right-angled triangle around the area for moving.

5.4 Execution of the Experiment

In this section we explain the execution of the experiment. First, probands are introduced to the study and then asked to fill out the page with personal information. Then probands are provided with an introduction to ExplorViz and learn the controls for virtual reality in a training phase. Following, the probands are asked to fulfil some tasks collaboratively. Lastly, the probands rate their user experience.

Introduction One of the probands uses the HTC Vive and the other one the Oculus Rift. The probands can choose on their own who of them uses which device. Firstly, the probands are greeted and asked to sit down and read an introducing text on the first page of the questionnaire. Here, it is explained that the participation in the study is voluntary. When both probands are finished, they are asked to fill out a page with personal information. After both probands are finished, they are asked to turn to the next page. This page contains an introduction to ExplorViz. This section can be skipped if the proband is already familiar with ExplorViz. If the probands speak German and wish so, the text can be read out loud and translated to German. When the text is read completely, the introduction is finished and the training phase starts.

Training phase

In preparation for the tasks that probands should work on collaboratively it is important to give an introduction to the HTC Vive (or Oculus Rift) and ExplorViz with the installed extensions. Therefore, we configure the backend with a software landscape which consists of one system and one application. This landscape is used to explain the controls. Therefore, a page with the assignment of keys and respective functions of the controllers is prepared

5.4. Execution of the Experiment

(see Appendix 7.2). However, we do not let the probands read this page on their own. In our opinion the best way to learn the controls is by practically showing them the features when they already wear the HMD and use controllers. Therefore, two tutors explain each proband the controls individually. This happens in parallel and in the same room but we suspect that the two tutors and probands do not disturb each other.

We give all probands a practical introduction to the following topics:

- ▷ Moving & Teleporting
- ▷ Menu interaction
- ▷ Moving the landscape
- ▷ Changing the camera height
- ▷ Changing the connection status
- ▷ Open and close elements of the landscape / application
- ▷ Move an application
- ▷ Show additional information about packages / classes
- ▷ Highlight closed components / classes
- ▷ Displaying the user list

We mention the spectating feature but do not show it because every proband should complete the training phase without any virtual interaction with the other proband. At the end of the training phase we check with our page which contains the controls to make sure that every feature is explained to every proband. When the explanation is finished for both probands we go over to the next phase.

Collaborative tasks

When both probands are familiar with the controls we restart the backend and configure it to show a larger software landscape with 6 systems. The probands are in the meantime allowed to keep on the HMDs. We explain them that they are about to fulfill tasks (see Appendix 7.2) to get familiar with the software. Most of the tasks are designed to be solved collaboratively. We point out that they are allowed and encouraged to talk to each other. Additionally, we explain that the assignment of keys is quite similar between the Oculus Rift and the HTC Vive.

Since the probands wear HMDs they are not able to read the tasks on their own. Thus, we read the tasks out loud and repeat the task if requested. Whenever controls are unclear we encourage them to speak to each other first but we explain controls again if necessary. When the probands fulfill the last task they are informed that they can lay the controllers on the table and remove their HMDs again.

5. Evaluation

Rating

Subsequent to the solving of tasks the probands are asked to sit down and fill out the rating pages including the text fields for further proposals and remarks. We shortly explain the structure of those pages and take a few steps back to let the probands fill out the pages without interruptions or stress. Probands are allowed to ask questions if any occur and we answer those briefly without coming closer or looking at their pages. When both probands signalize that they are finished we inform them that the experiment is thereby finished. We collect their pages and store them safely in a locked cabinet until we evaluate the results.

5.5 Results

In this section we present the results of the conducted usability experiment. First we present the accumulated personal data and then go over to the results for the statements which the probands rated. Since it is inappropriate to present each written statement of the probands here, the statements and all raw data of the evaluation can be found in Appendix 7.2.

Accumulation of data

In order to accumulate the collected data we assign integer values to the given answer in order to measure how much probands were agreeing with a statement. A positive value tells that a proband checked the box with the respective amount of pluses and therefore agrees with a statement. Accordingly, a negative value tells that, ignoring the algebraic sign, a proband checked the box with the respective amount of minuses and thus disagrees with a statement. Therefore, the results for the personal questions can range from 0 to 3. The results for the other questions can range from -2 to 2. -2 would mean that a proband did not agree at all with a statement and 2 that a proband fully agrees with a statement.

Personal Information

22 probands participated in our experiment. Thereof, 11 bachelor computer science (CS) students, 1 master CS student, 2 bachelor business informatics students, 3 students with another subject of study, 4 CS researchers and 1 proband with another profession. 19 probands were female and 3 male. In average the probands were 25.9 years old. 8 probands were wearing glasses. The remaining results of the personal questionnaire are listed below in Table 5.9.

Table 5.9. Results of the personal questions

ID	Mean	SD
A3	2.0	0.9258
A4	0.3636	0.9021
A5	0.7273	1.0320
A6	0.0455	0.2132
A7	0.9091	1.0193
A8	0.4545	0.7385
A9	1.9091	0.8679

Interaction

We developed or altered features to move in the virtual world or interact with the software models and a menu. Therefore, we asked the probands to rate certain general aspects. The results for this are shown in Table 5.10.

Table 5.10. Results of the interaction statements

ID	Mean	SD
B1.1	1.136	1.246
B1.2	1.773	0.429
B1.3	1.727	0.703
B1.4	1.455	0.596
B1.5	1.591	0.908

General Usability

To gain information if the developed VR features are in general suitable to work collaboratively with in VR, we asked the probands to rate how much they agree to a statement. The results are listed in Table 5.11.

5. Evaluation

Table 5.11. Results of the general statements

ID	Mean	SD
B2.1	1.909	0.294
B2.2	1.545	0.739
B2.3	-1.773	0.685
B2.4	1.364	1.093

Visual Appearance

As most of our work is concentrated in the frontend extension, we asked the users to rate certain aspects of the visual appearance. The results for this category are displayed in Table 5.12.

Table 5.12. Results of the visual appearance statements

ID	Mean	SD
B3.1	1.545	0.510
B3.2	1.136	0.990
B3.3	1.682	0.477
B3.4	1.727	0.703
B3.5	1.045	0.950
B3.6	1.045	1.133

Spectating

We are also interested in how well our approach for spectating other users is suitable for working collaboratively in VR. The corresponding results are listed in Table 5.13.

Table 5.13. Results of the spectating statements

ID	Mean	SD
B4.1	-0.773	1.412
B4.2	0.682	1.460

5.6 Discussion

In this section we discuss the presented results. Therefore, we correlate the results with statements of the probands or our observations.

Personal Information

Most probands stated to have good or very good knowledge of objectoriented programming. This is not surprising because 18 of 22 probands had a background in computer science and therefore objectoriented programming is likely to be taught or even self taught.

Only the participating researchers and none of the other probands had experience with ExplorViz. This possibly indicates that software visualization is not yet a much requested technology for computer science students. More than half of the probands have no experience with virtual reality and the experiences among the other participants are varying a lot. We suspect more people will have experience with virtual reality when the purchasing costs are further reducing in the future.

Only one proband stated to be slightly claustrophobic, hence we avoid correlating this data with other results. Being afraid heights was much more common among the probands than being affected by seasickness. Regarding this it could be that some probands do not yet know that they are in fact suffering from seasickness.

Every proband stated to know the other proband. Thus, we can not make any statements on the collaborative use for our technology for strangers. However, we assume that for most use cases of our technology the users are colleagues at work.

General Usability

All probands had the impression to be in the same room with the other user in virtual reality. Only 2 probands rated this statement with + instead of ++. We therefore conclude that the synchronization of the views works well. The majority of probands thought that the VR extension for ExplorViz is suitable for team work. Only one proband rated this statement with a minus. In our opinion the ability to see what other users are looking and the ability to point at objects with the controllers greatly helps for team work.

The feel of nausea or alike was rated with -1.773 during the experiment (except spectating) which means almost no nausea occurred. In our experience nausea during VR can occur whenever the movement in virtual reality does not match the movement in the real world. This only happens with our approach when a user teleports and therefore only for a fraction of a second the movement is decoupled from the real world. Overall most users would use ExplorViz with VR again. Thus, we conclude that the general usability is good.

Interaction

Moving and rotating the landscape received the most critical feedback in the topic of interaction. One reason is that the landscape can only be moved and rotated with many single clicks. This is not very comfortable when the position of the landscape should be

5. Evaluation

altered a lot. Another reason is that moving the landscape and moving an application is implemented very differently. The movement of the application was rated as very intuitive. Therefore, it occurred that probands tried to grab the landscape just like they could grab an application.

One obvious solution would be to allow the landscape to be moved just like applications can be moved. We consider to try this approach but in our opinion this could lead to conflicts when many people are interacting with the landscape. Certainly, the usability can be improved by enabling to move the landscape via the trackpad or with virtual analog sticks. This would allow for a smooth movement instead of many single clicks that move the landscape step-wise. In addition to altering the movement options for the landscape we have got the impression it would be reasonable to add a button to the landscape menu to restore the initial position of the landscape easily.

The movement through the virtual space was rated as intuitive. The same is true for the navigation through the menu. However, we would consider to implement the additional option to move with help of the trackpad on the left controller through menu. This would allow a one-handed use of the menu instead of using both controllers.

Lastly most probands agree that highlighting objects is a useful feature. One proband mentioned that pointing with the ray of the controller on an object is more intuitive. Another proband would have liked the option to highlight multiple objects. In our opinion highlighting multiple objects could lead to confusion, especially if many users are using this feature.

We noticed that some probands were confused that closed components can not be highlighted. To provide a better usability we plan on enabling this functionality.

Visual Appearance

The structure of the menu, movement of other users and number of text insertions was rated very positively. The visualization of other players was mostly positive but some probands would have preferred a complete (upper) body instead of only the HMD and controllers as visualization. We are concerned that especially with many users a more complete representation could lead to users each others sights. To avoid this problem an option could be added that users can choose locally via the menu how other users should be represented for them.

One proband mentioned that text overlays should contain larger text and another one that the text was not readable at the edges of the display. In general most users had little problem reading the hints. However, small text labels for classes or components were mentioned to be barely readily by many probands. One issue is the display resolution of the HMDs. Still, many text labels are very small and thus a feature to enlarge text when a controller points at it would be desirable.

Spectating

The statements about the spectating features received the most mixed feedback. Most

people did not feel any nausea but 7 probands either rated '+' or '++' here. It is striking that all of those 7 probands also stated to be afraid of heights. The other way around only 5 of the remaining 15 probands were afraid of heights, too. Additionally, the 7 probands who felt nausea or alike were more afraid of heights than the 5 remaining people who also were afraid of heights. As the sample size is still quite small we do not want to emphasize this correlation too much. However, the loss of control during the spectating could be especially uncomfortable for people who are afraid of heights. This possible connection should be further researched in future studies.

Unsurprisingly, people who felt nausea or alike stated that they would not use the spectating feature again. An additional factor for the feedback on statement B4.2 is that the use cases for the spectate feature were not well motivated by the executed tasks during the experiment. The probands were standing and had enough area to move around when they used the spectating feature. In our opinion spectating could potentially be used by people who sit and do not have much space to move around.

5.7 Threats to Validity

All statements which the probands should rate were phrased positively. Thus it could be that the probands were more driven to agree with the positively phrased statements. Then again the probands very clearly disagreed with statement B2.3. Additionally, it is our primary goal to report tendencies in this evaluation of usability.

Our probands were mostly students, the remaining probands were researchers or people without a background in computer science. Therefore, the evaluation might give insights in how well our technology can be used by people who mostly have not much prior knowledge of software visualization. However, the evaluation is missing probands with a professional industry background like software architects. As a result this evaluation can not clearly answer the question how well professional users would rate the current state of development. This is an important question because professionals are the target audience for a software like ExplorViz.

The evaluation was conducted under mostly ideal conditions. The computers had no performance issues and were connected via a fast and reliable LAN network. The probands also had enough space to walk and it was achieved 360 degree tracking of the HMDs and controllers without any sensor objects that could possibly block the sensors. It should be noted that at least the network connection speed and delay do not reflect the typical use case scenario we designed the software for.

5.8 Summary

Summarized, the probands gave very positive feedback. 22 probands conducted our experiment in groups of two. The movement options were perceived as intuitive and the

5. Evaluation

developed extensions were rated as suitable for teamwork. Only the readability of small text labels and the spectating feature received a mixed feedback. The raw data of our evaluation can found in Appendix 7.2. Even though it was our goal to request tendencies, we conclude due to the evaluation that our approach for collaborative software exploration in VR is promising and should be followed up on.

Related Work

In this chapter we present related work and compare other approaches to the one we chose. However, for all we know there are no scientific publications which cover the use of collaborative VR for software which is similar to ExplorViz. Instead, we take a look at collaborative approaches for software monitoring and approaches for the use of VR for software exploration.

[Fittkau et al. 2015a] research the practical use of physical 3D models. Fittkau et al. use ExplorViz to export models and print those with help of a 3D-printer. In order to print a larger software model, twelve smaller parts are printed and then glued together. Since a monochromatic printer is used, the models need to be painted in a time-consuming effort. More modern but also more expensive hardware would simplify these steps.

Fittkau et al. conducted an experiment using pairs of probands who should solve given tasks, too. However, Fittkau et. al also try to compare the use of physical models with collaborative work in front of a monitor. Even though 112 probands participated, the results are diverging due to the assigned tasks which can hardly be compared between those approaches. Still, it is assumed that the ability to point at certain parts of software is an advantage in terms of usability and efficiency.

Concluding, using 3D-printed models is the closest approach to ours in terms of a collaborative use of ExplorViz.

[Misiak et al. 2018] developed a technology called IslandViz to explore modular software systems in virtual reality. IslandViz can be used with the HTC Vive, too. As opposed to the city metaphor which ExplorViz employs, IslandViz uses an island metaphor. Therefore, bundles are represented as island, packages as regions on those islands and individual classes are represented as buildings on those islands. To visualize dependencies, islands can be connected via ports and employed arrows connecting those ports.

Also, in IslandViz a user can move through a virtual room. However, the software models cannot be moved freely throughout the room. In the middle of the room a virtual table is placed on which a virtual map is shown. This map represents a three-dimensional hologram. The map allows for interaction methods like zooming and translation with help of the controllers. Through this, [Misiak et al. 2018] want to combine different abstraction

6. Related Work

layers in one model and allow users to switch abstraction levels by zooming in or out. ExplorViz employs different abstraction layers by using two different views and nesting components on a visual level.

Since IslandViz currently can not be used collaboratively, there are no mechanism implemented which allow the use with multiple users. Concluding, IslandViz uses a different metaphor to visualize objects and software models are contained in a virtual map. Other than that, IslandViz is using a similar approach to allow users exploring software landscapes in virtual reality.

Conclusions and Future Work

In this chapter we, conclude our thesis and describe potential future work.

7.1 Conclusions

In this thesis we presented an approach to use the software visualization tool ExplorViz collaboratively in virtual reality. We used the HTC Vive as a fundamental hardware and developed a concept to extend ExplorViz. To do so we designed a new extension for the backend and built on top of an existing frontend extension. Proceeding from the concept we presented details of the implementation. In addition to synchronized models we introduced additional interaction tools for the user. Among other features a menu, text messages and a representation for other users were introduced.

To validate the usability of our approach we conducted a usability experiment with 22 probands. The results indicate an overall good usability. The movement through the virtual room and interaction with models were rated positively. The newly developed menu was intuitive to use and the current state of development is suitable to be used for team work. It is important to note that the use of virtual reality had no negative effect like nausea to the probands. Only the feature which allows to spectate other users received mixed feedback. Concluding, collaborative virtual reality is a promising technology for exploring software landscapes.

7.2 Future Work

We have shown that the collaborative use of virtual reality is a promising addition to ExplorViz. As a next step an evaluation with more probands could lead to further insights into the practical use of collaborative VR. For that evaluation it would be desirable to have at least 50 participants or respectively 25 participating groups of two to gain more reliable information.

Since some participants of the evaluation had problems reading smaller text labels it would be interesting to research whether improved hardware would lead to an overall better readability of text and thus improve usability. The displays of the HTC Vive Pro¹, a

¹<https://www.vive.com/de/product/vive-pro/>

7. Conclusions and Future Work

successor of the HTC Vive, have 78% more pixels than those of the HTC Vive. Furthermore, there is a wireless adapter for the HTC Vive Pro available which allows the use of VR without interfering cables. The HTC Vive Pro is already available for purchase and should work with the existing software without any bigger issues. Thus, we recommend a hardware upgrade to research potential usability improvements.

In the current state it is not possible to add users dynamically, change passwords or have any personal data like an email address associated with an account. For this reason we suggest to include a user management system in the backend and add a new tab to the web interface to edit the own and in case of an administrator also other user's account information.

The long-term goal should be to add support for augmented reality technologies in ExplorViz. Augmented reality incorporates virtual graphics into the real world. ExplorViz would make good use of this technology because ExplorViz only contains models which intuitively could be placed on floors, tables or at walls to work with them. One prominent hardware device for augmented reality is the Microsoft HoloLens². One drawback of the HoloLens is that only a limited part of the user's field of view can be enriched with augmented reality. Future hardware could have a bigger field of view and thus be a well suited addition to virtual reality in ExplorViz.

²<https://www.microsoft.com/de-de/hololens>

Bibliography

- [Billinghurst et al. 2001] M. Billinghurst, H. Kato, and I. Poupyrev. The MagicBook - moving seamlessly between reality and virtuality. *IEEE Computer Graphics and Applications* 21.3 (May 2001), pages 6–8. (Cited on page 5)
- [Caserta et al. 2011] P. Caserta, O. Zendra, and D. Bodénès. 3D Hierarchical Edge Bundles to Visualize Relations in a Software City Metaphor. In: *6th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2011)*. Williamsburg, United States, Sept. 2011. (Cited on page 7)
- [Fittkau et al. 2015a] F. Fittkau, E. Koppenhagen, and W. Hasselbring. Research Perspective on Supporting Software Engineering via Physical 3D Models. In: *IEEE 3rd Working Conference on Software Visualization (VISSOFT 2015)*. IEEE, Sept. 2015, pages 125–129. (Cited on pages 2, 43)
- [Fittkau et al. 2015b] F. Fittkau, A. Krause, and W. Hasselbring. Exploring Software Cities in Virtual Reality. In: *IEEE 3rd Working Conference on Software Visualization (VISSOFT 2015)*. IEEE, Sept. 2015, pages 130–134. (Cited on page 1)
- [Fittkau et al. 2017] F. Fittkau, A. Krause, and W. Hasselbring. Software landscape and application visualization for system comprehension with explorviz. *Information and Software Technology* 87 (2017), pages 259–277. (Cited on page 6)
- [Fittkau et al. 2015c] F. Fittkau, S. Roth, and W. Hasselbring. ExplorViz: Visual Runtime Behavior Analysis of Enterprise Application Landscapes. In: *23rd European Conference on Information Systems (ECIS 2015)*. Mai 2015. (Cited on page 6)
- [Fittkau et al. 2013] F. Fittkau, J. Waller, C. Wulf, and W. Hasselbring. Live Trace Visualization for Comprehending Large Software Landscapes: The ExplorViz Approach. In: *1st IEEE International Working Conference on Software Visualization (VISSOFT 2013)*. Sept. 2013, pages 1–4. (Cited on page 1)
- [Häsemeyer 2017] T. Häsemeyer. Kollaboratives Erkunden von Software mithilfe virtueller Realität in ExplorViz. Bachelorarbeit. Kiel University, Sept. 2017. (Cited on pages 1, 2, 7, 14)
- [König 2018] D. König. Collaborative Software Exploration with the Oculus Rift in ExplorViz. Bachelorarbeit. Kiel University, Sept. 2018. (Cited on page 1)
- [Liu and Sun 2012] Q. Liu and X. Sun. Research of web real-time communication based on web socket. *International Journal of Communications* 5.12 (2012), pages 797–801. (Cited on page 12)

Bibliography

- [Misiak et al. 2018] M. Misiak, A. Schreiber, A. Fuhrmann, S. Zur, D. Seider, and L. Nafeie. IslandViz: A Tool for Visualizing Modular Software Systems in Virtual Reality. In: *6th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2018)*. Köln, Germany, 2018. (Cited on page 43)
- [Misue et al. 1995] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages & Computing* 6.2 (1995), pages 183–210. (Cited on page 14)
- [Ohzawa 1998] I. Ohzawa. Mechanisms of stereoscopic vision: the disparity energy model. *Current Opinion in Neurobiology* 8.4 (1998), pages 509–515. (Cited on page 5)
- [Roberts and Wessler 1970] L. G. Roberts and B. D. Wessler. Computer Network Development to Achieve Resource Sharing. In: *Proceedings of the May 5-7, 1970, Spring Joint Computer Conference*. AFIPS '70 (Spring). Atlantic City, New Jersey: ACM, 1970, pages 543–549. (Cited on page 12)
- [Robertson et al. 1997] G. Robertson, M. Czerwinski, and M. van Dantzich. Immersion in Desktop Virtual Reality. In: *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*. UIST '97. Banff, Alberta, Canada: ACM, 1997, pages 11–19. (Cited on page 5)
- [Sutherland 1968] I. E. Sutherland. A Head-mounted Three Dimensional Display. In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*. AFIPS '68 (Fall, part I). San Francisco, California: ACM, 1968, pages 757–764. (Cited on page 5)
- [Zirkelbach et al. 2018] C. Zirkelbach, A. Krause, and W. Hasselbring. On the Modernization of ExplorViz towards a Microservice Architecture. In: *Combined Proceedings of the Workshops of the German Software Engineering Conference 2018*. Volume Online Proceedings for Scientific Conferences and Workshops. Ulm, Germany: CEUR Workshop Proceedings, Feb. 2018. (Cited on pages 6, 11, 12)

Appendix A

Dear participant,

we thank you very much for your participation in this experiment. You and another participant are about to test and evaluate a collaborative virtual reality (VR) extension to ExplorViz. You are free to quit the evaluation whenever you are not feeling comfortable anymore. First of all, we would like you to answer some general questions about you on the next page. This will allow us to add some context to the results later on. Your answers and test results are anonymous. Then we are proceeding by introducing you to ExplorViz and the input devices for VR. When you have learned the basics about ExplorViz and the controls you will be asked to solve some exercises together with your partner. Lastly, you will be asked to rate certain aspects about the user experience. This will help us improve the extension and tell us how well ExplorViz can be used with multiple users in virtual reality.

1 General Personal Data

ID: *Vive* ☐ *Rift* ☐

Age:

Profession: *student* ☐ *researcher* ☐ *other* ☐

Subject of study: *bachelor* ☐ *master* ☐

Gender: *male* ☐ *female* ☐ *diverse* ☐

Do you wear glasses? *yes* ☐ *no* ☐

Do you have any visual impairment? *yes* ☐ *no* ☐

If so, which:

	0	+	++	++ +
Experience with objectoriented programming	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Experience with ExplorViz	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Experience with VR	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Are you claustrophobic?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Are you afraid of heights?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Do you suffer from seasickness?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
How well do you know the other proband?	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

2 Introduction

ExplorViz is a monitoring and visualization software for large software landscapes. ExplorViz uses two different views for the visualization which are shown simultaneously with the VR extension. The landscape view is a view of a software landscape and is particularly suitable to get an overview of landscapes. Here you can see systems (grey), servers (green) and the software running on the servers (blue). The communication between software is represented by orange lines, where the thickness of the lines correlates with the number of calls it represents.

The application view represents a three-dimensional model of the software. On top of a grey foundation software packages (components) are shown in green, which in turn can contain components or individual classes (blue). The height of the blue blocks indicates the number of objects belonging to the class. Here, too, the communication between objects is visualized with orange lines. You can select individual classes or call up additional information for a class.

This type of representation is intended to be a metaphor for a three-dimensional city, with the classes here representing (high-)buildings and communication between classes are streets.

Now that you are familiar with the concepts we would like you to get familiar with the controls. On the next page there is an overview of the functionalities of all buttons on the controllers you are about to use. Please use the HTC Vive or Oculus Rift respectively now while we guide you through all control options. Feel free to ask questions throughout the experiment if something is unclear to you.

3 Tasks

1. Connect via the menu.
2. Change your height as you like.
3. Open the user list.
4. Move and rotate the landscape as you like.
5. Find the node '10.0.2.2'.
6. Find the application 'Wiki' and try to open it.
7. With how many applications does 'Webshop' communicate?
8. Open the application 'Webshop'.
9. Move and rotate the application 'Webshop' as you like.
10. Mark the class 'ItemHelper' which is part of the application 'Webshop'.
11. How many active instances has the class 'ImplementationHandler'
(located in component org/webshop/kernel/impl)?
12. Open a second application.
13. Use the spectate feature (only as long as you are comfortable using it).
14. The spectated proaband may mark the component labeling in Webshop.
15. Quit spectating and spectate each other with reversed roles.
16. Mark component connector in DatabaseConnector.
17. Quit spectating.
18. Close all open system and components.
19. Disconnect via the menu.

4 Rating

	--	-	+	++
Moving and rotating the landscape is well realized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
The moving and rotating of the applications is well realized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
The movement (incl. teleportation) in the virtual space is intuitive	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
The structure of the menu is intuitive	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
Navigation through the menu is intuitive	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
The visualization of other users is well done	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
The movements of the other user were displayed to me without delay	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
I had the impression that I was in the same room with the other user (positions and state of users, landscape & applications were synchronized)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
The number of text insertions was reasonable	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				

	--	-	+	++
Text overlays were clearly readable	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
Text overlays were visually appealing (duration, length, position, color, size, animation)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
Highlighting objects is a useful feature	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
ExplorViz with VR extension is suitable for team work	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
During the experiment (except spectator mode) I felt nausea or something alike	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
While spectating I felt nausea or something of the like	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
I would use the spectator mode again	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				
I would use ExplorViz with the VR extension again	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
.....				

Proposals for future features

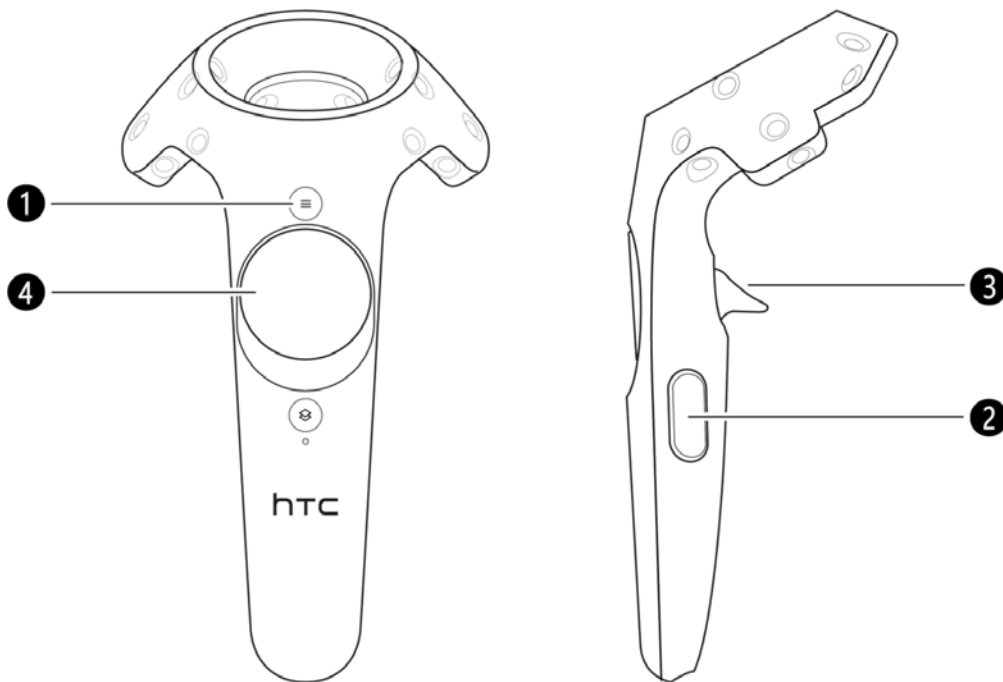
Improvement proposals

Further remarks

Appendix B

Controls

↳ Vive Controllers:



You can target many objects in the virtual environment with the ray of the controller and interact with them through corresponding buttons. The ray of the left controller is colored black and that of the right one is colored green.

❶: (Left Controller):

Press this button to open the options menu. If in a menu, pressing the button can be used to navigate back through previous menus.

❷: (Left Controller):

Hold this button down to display a list of users connected to the server. Release the button to close the list.

❸: (Right Controller):

Target a 3D application with the ray of the controller and keep this button pressed to bind the 3D application to the controller. The application now follows all movements of the controller. Release the button to stop this behavior.

❹: (Left Controller):

Target the ground with the ray of the left controller and press this button to teleport yourself to the displayed circle on the ground. Target the red "X" above a 3D application with the ray of the controller and press this button to delete the 3D application. This button can also be used to select targeted clazzes and closed packages of a 3D application. Consequently the selected entity is colored red and the associated communication lines are highlighted. If nothing is targeted press this button again to unselect the entity and restore its color and the communication lines.









③: (Right Controller):

Press this button to open/close targeted systems, nodegroups, packages and create 3D applications out of targeted 2D applications. Target the red "X" above a 3D application with the ray of the controller and press this button to delete the 3D application. This button can also be used to navigate through menus.

④: (Right Controller):

Press this button to display information about the targeted entity.

Keyboard:

- : Move the camera upwards
- : Move the camera downwards
- : Move the camera leftwards
- : Move the camera rightwards
- : Move camera forwards (Zoom in)
- : Move camera backward (Zoom out)
- : Rotate the environment forwards
- : Rotate the environment backwards

Appendix C

ID	Vive/Rift	Age	Profession	Subject	Gender	Glasses?	Visual Impairment	Which?	OO	ExplorViz	VR	Claustro	Heights	Seasickness	Well know
1	Vive	24	student	student teacher	female	no	no		0	0	0	0	0	0	3
1	Rift	23	student	Biology	female	yes	yes	short	0	0	1	0	0	1	3
2	Vive	32	researcher	CS	male	yes	yes	short	2	3	2	0	2	2	2
2	Rift	28	researcher	CS	male	yes	yes	red/green	2	3	2	0	2	2	3
3	Vive	54	researcher	CS	male	no	no		2	1	1	0	0	0	2
3	Rift	39	researcher	CS	male	yes	yes	short	3	1	0	0	1	0	2
4	Vive	20	student	CS BA	male	no	yes	slight red/green	2	0	0	0	0	0	1
4	Rift	24	student	CS BA	male	yes	yes	short	2	0	0	0	1	0	1
5	Vive	22	student	CS BA	male	no	no		3	0	0	0	0	0	1
5	Rift	student	student	CS BA	male	no	no		2	0	1	0	2	0	1
6	Vive	22	student	CS BA	male	yes	yes	short	3	0	0	0	1	0	1
6	Rift	22	student	business CS BA	male	no	no		2	0	0	0	0	1	1
7	Vive	28	student	business CS BA	male	no	no		3	0	0	0	0	0	1
7	Rift	21	student	CS BA	male	no	no		3	0	3	0	1	1	1
8	Vive	22	student	CS BA	male	yes	yes	short	2	0	2	0	3	0	3
8	Rift	19	student	CS BA	female	yes	yes	long	2	0	0	0	0	0	3
9	Vive	22	student	CS BA	male	no	no		2	0	0	1	1	0	2
9	Rift	22	student	CS BA	male	no	no		2	0	0	0	0	0	1
10	Vive	22	student	CS BA	male	no	no		3	0	1	0	0	1	3
10	Rift	22	student	CS MA	male	no	no		2	0	3	0	1	0	3
11	Vive	30	other		male	yes	yes	short, 90% stereoscopic vision	2	0	0	0	2	2	2
11	Rift	26	student	history / philosophy BA	male	no	no		0	0	0	0	3	0	2
AVERAGE		25.9048							2.0000	0.3636	0.7273	0.0455	0.9091	0.4545	1.9091
SD		7.9618							0.9258	0.9021	1.0320	0.2132	1.0193	0.7385	0.8679

	#On Paper:	1	2	3	5	12	8	13	14	17	4	6	7	9	10	11	15	16
ID	Rift/Vive	B1.1	B1.2	B1.3	B1.4	B1.5	B2.1	B2.2	B2.3	B2.4	B3.1	B3.2	B3.3	B3.4	B3.5	B3.6	B4.1	B4.2
	1 Vive	2	2	2	1	2	2	2	-2	2	1	2	2	2	1	0	-2	2
	1 Rift	2	2	2	1	1	2	1	-2	2	1	1	1	2	1	1	-1	2
	2 Vive	1	1	2	2	2	2	2	-2	2	2	1	1	2	1	1	1	1
	2 Rift	2	2	2	2	2	2	-1	-2	-1	2	2	2	2	-1	2	1	1
	3 Vive	2	2	2	2	1	2	2	-2	2	1	1	2	2	2	2	-2	2
	3 Rift	2	1	2	1	2	2	2	-1	2	1	-1	2	2	2	-1	-2	
	4 Vive	1	2	1	1	2	2	2	-2	1	1	2	1	2	1	2	-2	1
	4 Rift	2	2	2	1	-2	2	1	-2	-1	1	-1	1	1	1	1	-2	-2
	5 Vive	2	1	1	2	1	2	1	-2	1	1	2	2	1	2	1	-1	-2
	5 Rift	1	2	2	2	1	2	2	-2	2	2	1	2	2	1	2	1	1
	6 Vive	2	2	1	1	1	2	1	-2	2	2	1	2	2	1	2	1	1
	6 Rift	1	2	2	1	2	2	2	-1	2	1	1	1	2	1	1	-1	1
	7 Vive	1	1	2	1	2	2	1	-2	1	1	1	2	1	1	-1	-1	1
	7 Rift	-2	2	2	2	2	2	2	1	2	2	1	2	2	2	-1	1	-2
	8 Vive	-1	2	2	2	2	2	2	-2	2	2	-1	1	-1	-1	-1	1	-2
	8 Rift	2	2	2	2	2	1	2	-2	2	2	2	1	2	-1	2	-2	1
	9 Vive	2	2	2	2	2	2	2	-2	2	2	1	2	2	2	2	-2	2
	9 Rift	2	2	2	0	2	2	1	-2	0	1	1	2	2	2	2	-1	2
	10 Vive	1	2	2	2	2	2	2	-2	2	2	2	2	2	1	1	-2	2
	10 Rift	2	1	-1	1	2	2	1	-2	-1	2	2	2	2	1	2	-2	2
	11 Vive	-1	2	2	2	2	2	2	-2	2	2	2	2	2	2	2	-2	1
	11 Rift	-1	2	2	1	2	1	2	-2	2	2	2	2	2	1	1	2	-1
Rift		1.1818	1.8182	1.7273	1.2727	1.4545	1.8182	1.3636	-1.5455	1.0000	1.5455	1.0000	1.6364	1.9091	0.9091	1.0909	-0.5455	0.5455
Vive		1.0909	1.7273	1.7273	1.6364	1.7273	2.0000	1.7273	-2.0000	1.7273	1.5455	1.2727	1.7273	1.5455	1.1818	1.0000	-1.0000	0.8182
AVERAGE		1.1364	1.7727	1.7273	1.4545	1.5909	1.9091	1.5455	-1.7727	1.3636	1.5455	1.1364	1.6818	1.7273	1.0455	1.0455	-0.7727	0.6818
SDEV		1.2458	0.4289	0.7025	0.5958	0.9081	0.2942	0.7385	0.6853	1.0931	0.5096	0.9902	0.4767	0.7025	0.9501	1.1329	1.4119	1.4601

ID	Rift/Vive	Remarks
1	Vive	C2: ansprechenderer Hintergrund
1	Rift	C1: bessere Darstellung *stick person drawing* <- ungefähr so oder eine Figur nach Wahl z.B. Firmenmaskottchen C2: bessere Farben; schönerer Hintergrund
2	Vive	B3.6: overlay anchor different sometimes (menu vs. connected) C1: menu option: showing controller mapping (half-transparent) (help dialogue) C2: + readability of labels (components, classes, systems) + movement reset capability (via menu) + rotation of applications could improved C3: The VR-extension provides a "real" working together in the same room feeling, well done.
2	Rift	B1.1: würde gerne Controll-Stick ebenfalls für Rotation nutzen B3.2: show text field of other users? B3.5: Manche waren sehr am Rand und da konnte ich sie schwer lesen, z.B. 'no data' bei der Applikation C1: Rotation der Applikation mit dem Controllstick C2: Labels von Applikationen etc. besser lesbar machen C3: Sehr flüssig und insgesamt intuitiv. Aber vermutlich nicht für den echten Einsatz geeignet.
3	Vive	B1.3: nach einer gewissen Zeit B1.5: useful for what? B3.2: Stirchmännchen wäre gut C1: Avatar für Kollegen C2: "Durchwandern" der Stadt C3: Unterschiedliches Verschieben im Landscape & Application Modus
3	Rift	B3.2: AVATARS WOULD BE NICE B3.5: APPLICATION NAMES NOT ALWAYS READABLE
4	Vive	C1: red X to close application closer to the application and maybe smaller move landscape like the application spectate menu closable while spectating
4	Rift	B1.5: showing with lasers more intuitive B2.2: up to ca. 4 people B2.3: after a little B3.2: Give them a body B4.1: after a little C1: Schließen Button nach oben links Drehen mit Joystick dicker Laser (mit Laserschwertgeräusch beim kreuzen) Strand Umgebung Wasser \ Aqua
5	Vive	B3.1: spectator menü sollte man schließen können B4.1: spectator modus ist überflüssig, wenn man sich auch so etw. zeigen kann C2: - Klassen ausschreiben, nicht "impl..." - größere Schriften für kleine Klassen
5	Rift	
6	Vive	B1.5: Mehrfach highlighten noch besser

6	Rift	B1.4: Navigation mit Joystick ist einfacher B3.5: Brille war ein wenig unscharf B3.6: Zusätzliche Infos zu Klassen an Controller umständlich
7	Vive	C1: zoom in/out um text einer applikation usw. besser sehen zu können. momentan ruder ich sehr viel mit den Armen, um mir etwas ranzuziehen. (wahrscheinlich durch Teleport aber eigentlich gelöst) C2: Lag / Bilschirm "flackern" bei schließen von Komponenten sollte verbessert werden
7	Rift	B1.1: Genau wie Applikationen bewegen ist viel intuitiver B2.2: Ein Leiter, der verschieben kann, und Zuschauer, die Highlights, könnte bei mehr Leuten besser sein B2.3: Es wirkte so, als sei die Framerate instabil B3.6: Nicht wirklich aufgefallen B4.2: Hat keine wirklich nützliche Funktion C1: Darstellung als Kugeln, um Verbindungen besser sichtbar zu machen, im Gegensatz zu jetziger 2-dimensionaler Darstellung C2: Neigung von Menü und Info-Fenster etwas zum User hin bewegen
8	Vive	B4.2: Ich weiß nicht, wie sinnvoll der Spectator-Mode insgesamt ist. Im Prinzip ist man in der VR als Beobachter/in schon dabei und fühlt sich sowieso wohler aus eigener Perspektive
8	Rift	
9	Vive	C2: Blauen Kasten früher anzeigen um Mobilität zu erleichtern
9	Rift	B2.2: Um etwas zu zeigen auf jeden Fall, Kommunikation ist aber schwierig B4.1: Ungewohntes Gefühl, aber nicht besonders schlimm B4.2: Jedoch nur mit dem Grund, da es sich etwas unangenehm anfühlt C2: Den Coop-Partner als Körper darstellen, wie z.B. die Hände im "Eröffnungs-Raum" der Oculus
10	Vive	B3.5: teilweise unscharf oder zu klein (liegt vielleicht auch an der Brille) B3.6: Vorschlag: Text vergrößern, wenn man drauf zeigt C3: sehr nette Arbeitsatmosphäre
10	Rift	B1.2: Es wäre hilfreich Objekte aus der Entfernung greifen zu können B1.3: Keine selbstdrehen, keine kontinuierliche Bewegung B1.4: Man braucht ein bisschen viele einzelne Klicks B2.2: Wenn man den Platz und die Geräte hat schon B2.3: No B3.5: Manche waren etwas klein B3.6: Manche Label wurden abgeschnitten wenn sie zu lang waren B4.1: No B4.2: Yes C1: kontinuierliche Bewegung Objekte aus der Entfernung greifen Eine Suchfunktion C2: kleine labels an den Controllerbuttons (in VR) die sagen was der Button bewirkt
11	Vive	B1.1: Rotation fehlt C1: Personalisierung d. Hintergrunds, Menüfarben etc. C2: Highlights verschwinden bei Bewegung C3: Hat Spaß gemacht!

11	Rift	<p>B4.1: eye strain</p> <p>C1: animated connections to symbolize traffic or usage</p> <p>C2: spectator mode: highlighting should stay the same if the application is moved; applications should be able to be grabbed through the ground (or you shouldn't be able to go through the ground)</p>
----	------	--