Visualization Research Center of the University of Stuttgart (VISUS)

Bachelorarbeit

Exploring Interaction Modalities for Immersive Analytics and Situated Visualization

Niklas Gärtner

Course of Study:	Informatik
Examiner:	Prof. Dr. Daniel Weiskopf
Supervisor:	Dr. Leonel Merino
Commenced:	April 6, 2020
Completed:	November 24, 2020

Abstract

Although setups of immersive environments offer a vast quantity of interaction options, the typically used input devices lack the capability to perform complex interactions, such as creating text for programming, in a reliable, efficient, and mobile fashion.

As a result, developers who wish to create applications for immersive environments in domains like immersive analytics (IA) and situated visualization (SV) that feature complex interactions are forced to find workarounds to this problem, such as using a mobility constricting classical keyboard, inhibiting their design possibilities.

I replaced classical keyboards with an alternative – wearable keyboards, defining a domain-specific configuration for programming by creating a framework to map the inputs of wearable keyboard to the outputs required to easily create syntax of the programming language.

This solution enables users to perform this type of complex interaction while moving about without being forced to visually or at least spatially register position of the classical keyboards input options.

Kurzfassung

Obwohl es eine große Vielfalt an Interaktionsmöglichkeiten für immersive Anwendungen gibt, fehlt den üblicherweise genutzten Eingabegeräten die Fähigkeit komplexe Interaktionen – wie etwa das Erstellen von Text zum Programmieren – auf zuverlässige, effiziente und mobile Weise zu verrichten.

Als Folge sind die Designmöglichkeiten von Entwicklern, in immersiven Domänen mit potenziell komplexen Interaktionen wie „Immersive Analytics“ (IA) und „Situated Visualization“ (SV), eingeschränkt. Beispielsweise müssten sie dann auf die klassische, bewegungseinschränkende Tastatur ausweichen.

Ich habe die klassische Tastatur mit einer Alternative – einer tragbaren Tastatur ersetzt. Diesbezüglich habe ich für eine Domänen-spezifische Programmiersprache ein Framework implementiert, welches die Eingaben der tragbaren Tastatur zu benötigten Ausgaben wandelt, um einfach die Syntax der Programmiersprache zu erstellen.

Diese Lösung ermöglicht es Nutzern diese Art der komplexen Interaktion zu vollführen und sich währenddessen zu bewegen ohne auf die Position der Eingabemöglichkeiten der klassischen Tastatur optisch oder zumindest räumlich angewiesen zu sein.

Contents

List of Figures

List of Tables

Acronyms

AR Augmented Reality. 13

CAVE Cave Automatic Virtual Environment. 21

HCI Human Computer Interaction. 18

HMD Head-Mounted Display. 21

IA Immersive Analytics. 5

JSON JavaScript Object Notation. 30

SV Situated Visualization. 5

UI User Interface. 18

VR Virtual Reality. 13

1 Introduction

Virtual Reality (VR) was defined by LaValle [LaV20] as

> Inducing targeted behavior in an organism by using artificial sensory stimulation, while the organism has little or no awareness of the interference.

In their book [SH16], Schmalstieg and Höllerer mentioned a definition of Augmented Reality (AR) by Azuma. According to it AR has three main characteristics:

> *i*) Combines real and virtual *ii*) Interactive in real time *iii*) Registered in 3D

VR and AR represent avenues to display and explore interactive visualizations of complex data by exploiting benefits of such immersive environments. Up to date, many well-established visualization techniques, that proved effective when displayed in a computer screen, have been transferred to immersive environments [MBN18; MHB+19].

These novel immersive visualizations include IA and SV. IA can be loosely defined as the use of VR to display data visualizations. Similarly, the use of AR to display visualizations of data that relate to a specific location *in-situ* is called SV. When utilizing these types of immersive visualization the users' perception of the visualized data can be greatly different when compared to the non-immersive visualization on for example a computer screen. Immersive environments promote naturalness in their attempt to let users experience visualizations more similarly to how they would perceive them in the real world. This naturalness can be further improved by more natural interaction with the use of for example natural user interfaces (NUI). For instance, in them, users can navigate visualizations by walking and selecting elements through hand gestures. In this context devices with unique strengths like the computer keyboard for text generation can have a negative impact on naturalness. Mobility plays an important role in IA/SV. Therefore, IA/SV users need in-situ capabilities for authoring. I hypothesize that interaction modalities, specifically, wearable keyboards, could have a positive impact on users performance when authoring IA/SV.

Consequently, I formulated the following research questions:

RQ.1 What are the characteristics of interaction modalities employed in IA/SV?

RQ.2 How can wearable keyboards be used to support IA/SV authoring?

To address these questions, I followed two steps. Firstly, I designed and conducted a literature survey to analyze interaction modalities in IA/SV. Specifically, I collected 354 IA/SV results by querying main digital libraries. Of these, after a careful analysis, I included 45 papers that contain sufficient information to characterize interaction modalities. For each paper, I identified the used interactions including the human actions used to trigger the specific system reaction. Furthermore I collected given information about the used hardware and software. I synthesized the results into a taxonomy that encapsulates the properties of the interaction. Focus of the taxonomy is a categorization of interaction considering not only the semantics of the action in regards to its purpose but also the direct abstract object addressed by the action. The aim of this focal point is to reveal properties of naturalness in the designated types of interaction. In the results of my survey I found no use of devices that can be used as more natural alternatives for the generation of text than the traditional keyboard while possessing at least some of it's efficiency or reliability.

Secondly, I analyzed the feasibility and presented early results of using wearable keyboards for authoring IA/SV. That is, I designed and implemented a framework of a state machine to enable more complex interactions with a wearable chorded keyboard. In particular, I used the Tap Strap 2[1] device. In it, users type keys using chords. Specifically, the five fingers on the hand can be used to produce chords of 31 different keys. The whole set of possible inputs mapping to a specific group of outputs in this fashion can be regarded as a *mode*. The purpose of the state machine is the high number of possible modes since they are implemented as states. In this manner received inputs from the device in a particular state can be used to create specified outputs or transfer to another state. Thus, with one or more modes a mapping can be created that contains the aimed text content. As a use-case I created a mapping that features elements of a programming language syntax to potentially ease authoring with it in immersive environments. The essence of this mapping is using the default mapping of the wearable keyboard and expanding on it using not single characters but whole structures used by the syntax of the programming language.

[1]https://www.tapwithus.com/

2 Systematic Literature Review

Next, I present the literature review method employed to analyze interaction modalities used in IA/SV, and the results that I obtained.

2.1 Method

The systematic literature review method requires to define specific data sources and involve a reproducible search strategy accompanied with explicit criteria for inclusion and exclusion toward finding relevant papers. Specifically, I split RQ.1 into more fine-grained questions that I suspect can be answered through the results of the literature review.

RQ.1.1 What hardware is used in IA/SV?

RQ.1.2 What software infrastructures are involved in IA/SV interaction modalities?

RQ.1.3 What are the human actions involved in multiple interaction modalities?

In the following, I describe the employed process that I adapted from the process described by Kitchenham [KB13].

2.1.1 Data Sources and search strategy

To keep the number of potential studies used in this literature review manageable I only included papers that were published at either IEEE Xplore[1] or ACM DL[2]. To this end I used the search function of these digital libraries using the keywords "immersive analytics" and "situated visualization". The number of search results for IEEE Xplore were 122 for IA and 149 for SV. In contrast to IEEE Xplore for ACM DL I searched using these keywords in quotation marks getting 65 results for IA and 18 for SV. After filtering the studies based on the inclusion and exclusion process I started collecting studies using the forward and backward snowballing as described by Kitchenham. The number of studies passing the initial check-up exceeded 50 before completing the snowballing process of 10% of the already included studies. The predicted number of finally included studies when continuing would have far exceeded the aimed number. As a consequence of already having collected an expedient number of studies even when disregarding references and citations I excluded them.

[1]https://ieeexplore.ieee.org

[2]https://dl.acm.org

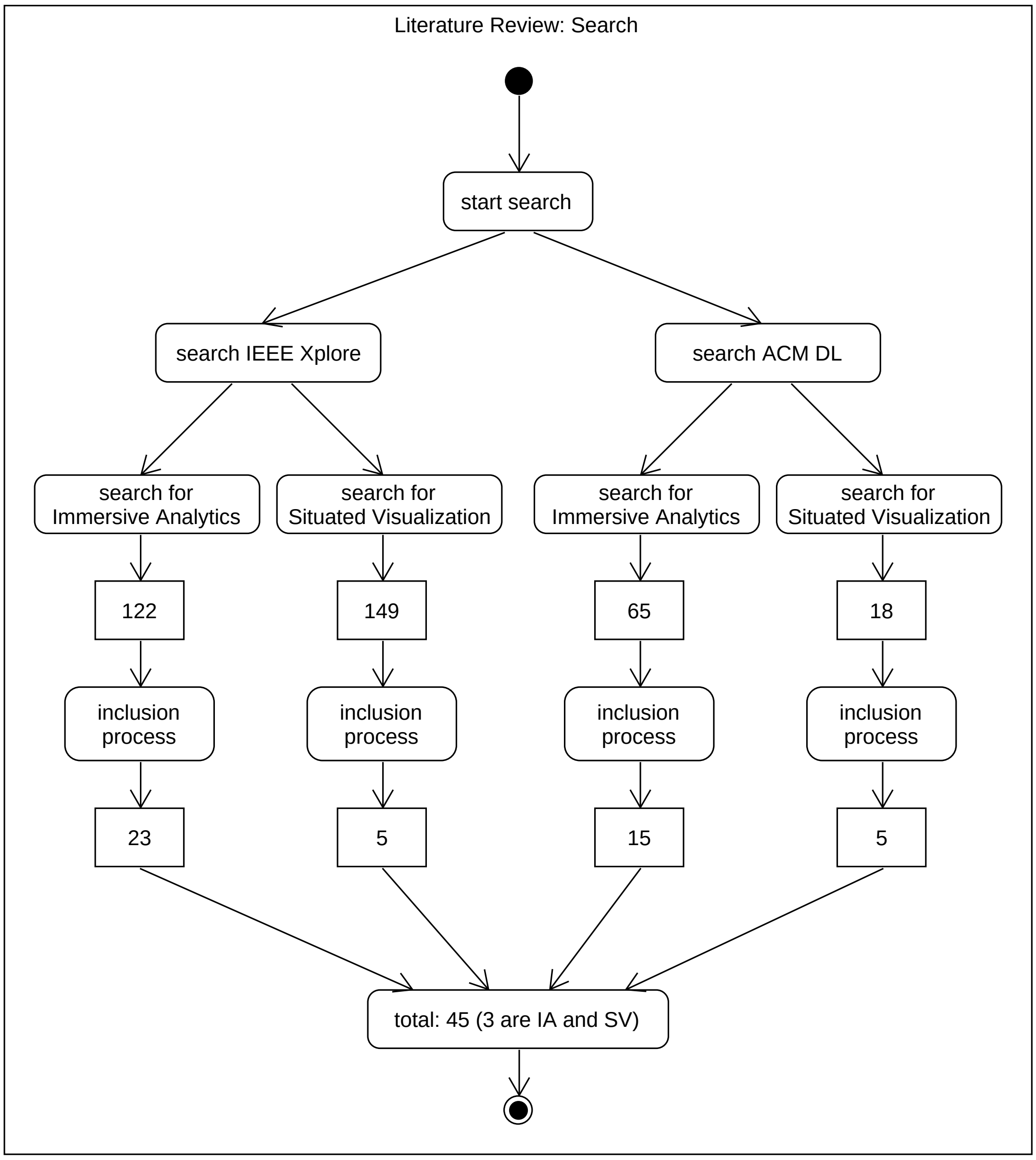

Literature Review: Search
start search
search IEEE Xplore
search ACM DL
search for Immersive Analytics
search for Situated Visualization
search for Immersive Analytics
search for Situated Visualization
122
149
65
18
inclusion process
inclusion process
inclusion process
inclusion process
23
5
15
5
total: 45 (3 are IA and SV)

Figure 2.1: A diagram visualizing the number of results at each step in the paper selection process

2.1.2 Inclusion and exclusion criteria

When analyzing the results returned by IEEE Xplore, I observed that some of them do not correspond to research papers. That is, results included documents that do not correspond to peer-reviewed IA/SV papers such as *Table of contents*, *Author index*, *Chairs and Committees*, *Copyright notice*, and *Sponsors*. Therefore, I excluded 12. From the remaining I included 87 papers that describe interaction modalities of IA/SV in sufficient detail. Lacking information concerning the question of by what action a certain interaction effect is achieved resulted in omitting said interaction within the study. In case no interaction was left in the study the whole paper was excluded, which was the case of 42 papers. Information of interaction was gathered only if a concrete and explicit implementation existed. Designs, research proposals, and tools' guidelines were excluded. However, prototypical example implementations of tools and frameworks for the demonstration purposes were included in case interactions were sufficiently detailed. Lacking information of hardware was a symptom of a design or proposal without an implementation. I only included IA/SV papers in which interactions take place in an immersive environment. All included studies are listed in Table A.1.

2.1.3 Data extraction

To find relevant data from papers that help me answer the formulated questions, I checked keywords and the abstract as a first step in order to get an idea of the topics involved in the paper. Next, I searched for information that could help me decide whether or not to exclude the paper.

I observed that there are particular paper sections that are more likely to contain specific data. Therefore, skimming through those sections was more effective than searching for generic keywords such as *interaction*, as this often lead to false positives. Indeed, a search like that often would match meta-information on interactions instead of a description of the actual application of an interaction. In cases where I still did not find interaction data, I read through the rest of the content of papers thoroughly to ensure not to miss important information. In rare cases keywords such as *hardware* were useful, however, often this method had a greater chance of false negatives. For example, if hardware was mentioned, it was usually to describe the used hardware involved in interactions.

2.1.4 Coding process

In order to classify the data extracted from papers I used a taxonomy. To define dimensions of the taxonomy that are meaningful for interaction modalities, I analyzed existing taxonomies presented in the information visualization research literature. The variations found in these taxonomies resulted from their different focus of attention and level of abstraction. The exploration of these taxonomies thereby consolidated my own decision for the categorization of interaction modalities.

When analyzing the literature, I observed that taxonomies adopt vastly different approaches. For the purpose of this research I looked at a few. Yi et al. [YKSJ07] presented a taxonomy that categorizes interactions based on the task that should be executed by it. The resulting categories they describe are *select*, *explore*, *reconfigure*, *encode*, *abstract/elaborate*, *filter* and *connect*. Brehmer and Munzner presented a taxonomy [BM13] that classifies interactions based on the intent of a visualization task through questions like *Why?*, *How?*, *What?*. Why? categorizes the purpose of the task into *consume* or *produce*. How? groups the way to achieve task completion into *encode*, *manipulate*, and *introduce*. What? splits interactions into *input* and *output* constraints. Additionally, I found taxonomies that classify interactions into more fine-grained categories. For instance, Nabiyouni et al. [NB16] presented a taxonomy that focuses on locomotion techniques in VR. This taxonomy categorizes walking-based locomotion depending on the properties used to achieve said action including *movement range*, *walking surface*, *transfer function*, *user support*, *walking movement style* and *input properties sensed* as well as several layers of sub-properties.

To classify the interaction modalities that I found in IA/SV papers, I used categories existing in previous taxonomies that exhibit an adequate level of granularity such as a taxonomy proposed by Shneiderman et al. [Shn96]. However, I also aimed to incorporate the value added by immersive environments in AR and VR (present in a lesser degree in traditional Human Computer Interaction (HCI)). I conjecture that interaction contributes to the feeling of immersion when it mimics natural interactions in the real-world. This naturalness of interaction is dependent on our expectancy of what a certain action does and is therefore reliant on affordances and constraints. In this manner the affordances and constraints dictate both the actions we take as well as the assumed purpose that action has. Figure 2.2 shows the thought process that I followed for this presumption. As attention was focused on the human interactions, I aimed to categorize them in a way that clarifies the semantic dependency the action has in regards to its purpose. To ease the comparison of categories, I attempted to balance the number of categories, and the number of cases a specific instance of a category occurs. Therefore, I considered factors that limit the number of categories and the number of category instances. My goal was to end up with an adequate number of instances. Limiting factors include merging semantically similar categories, merging rarely occurring categories, and splitting too large categories. For example, the category *Navigation* often used in other taxonomies was split into *Navigate View* and *Navigate Data*, in order to split these semantically different forms of navigation. The reason behind this decision is that, even though pure visuals may not be different, there is a semantic difference between *moving something* and *being moved*. Another example is the interaction category *Display Data* that would produce many instances as it is too specific to the hardware, software, and goals involved in the application. On the other hand, the interaction category *Save Data* is an example of a category that rarely occurs. I decided not to generalize it as the resulting category would have been too coarse to extract meaningful information. A taxonomy based on the semantics of interactions inevitably produces categories that enables many categories to be fulfilled at the same time. To avoid such cluttering, I selected the consequence of the highest abstraction level as active for the particular interaction. For example, activating a selection interactions through a User Interface (UI) is considered in the *Option Control* category, whereas selecting a UI item is included in the *Edit Selection* category. I observe that there is a potentially complex relationship between categories such as *Filtering* and *Details-on-demand*, as their classification could differ depending if considering either seeing all possible information or none as the default state. In the following, I list, describe, and explain the rationale behind the relevance of the categories I included in my taxonomy.

Aggregate. *Description:* Type of interaction that enable grouping based on aggregation type functions. These functions include creating, changing, or deleting a set or a part of a set. Employed methods are often merging, clustering, or slicing. *Relevance:* Semantic relevance of aggregating objects. It enables easier comprehension of complex data by grouping semantically similar data together. Can enables analyzing relations of data based on properties.

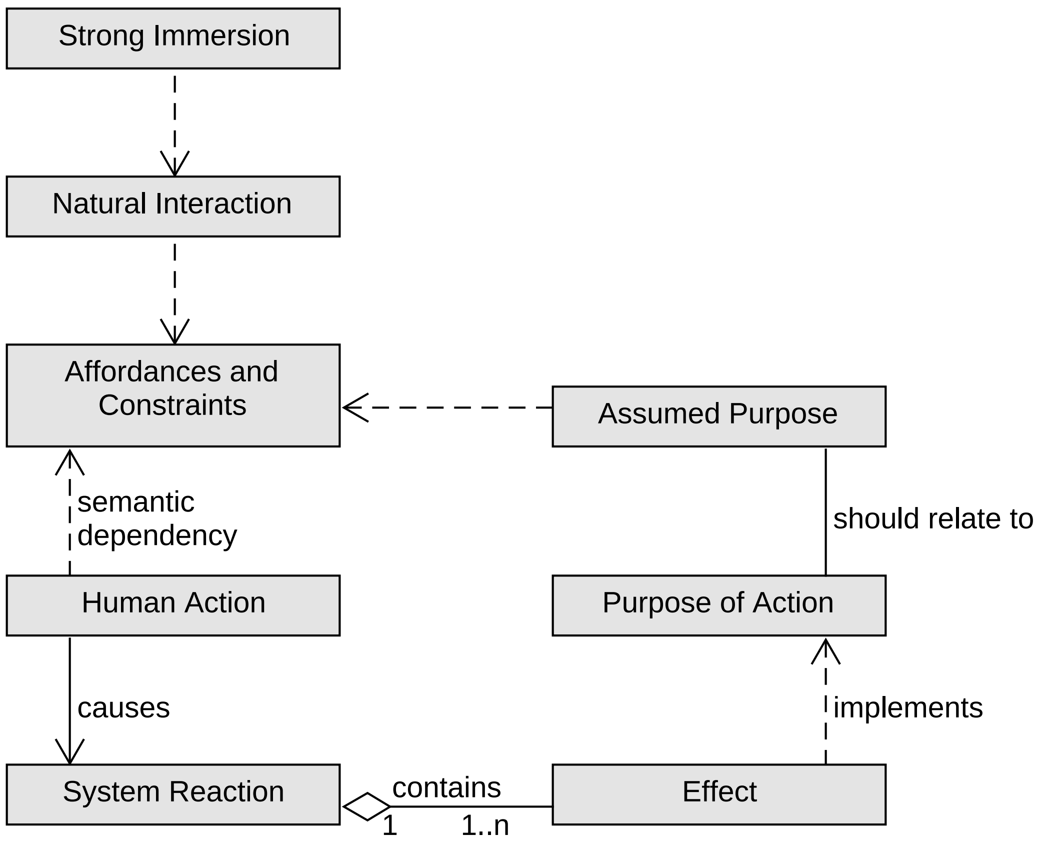


Figure 2.2: This visualization shows my assumptions of the relevance of natural interaction

Change modes. *Description:* Type of interaction that changes the effects induced by a specific set of inputs. For example, the action of pressing a specific button changes the system reaction of a specific set of input options. *Relevance:* Semantic relevance of changing input-effect relation. Enables artificial increasing the number of system reactions with a limited number of input options. Realizing semantically similar effects with similar inputs can increase learnablity.

Data scroll. *Description:* Type of interaction that enables scrolling through linearly arranged data / options. This does not include spatial navigation as it is already present in the categories Navigate data and Navigate view. In addition to data that can be spatially displayed and easily comprehended, there is also data with one or more dimension that does not make this possible. This includes data for which only a specific instance of its dimension can be comprehended at given moment such as time, in case all spatial dimension should be visible as well. Also included in this way is for example text with paragraphs containing different content. For such data up until now I only found implementations in which it was traversed linearly. In case an example is found where this is not the case this category has to be renamed or reworked. *Relevance:* Semantic relevance of manual linear maneuvering through data. Enables miscellaneous data navigation through other dimensions such as time.

Details-on-demand. *Description:* Type of interaction that reveals extra information to the user. Information is shown to the user when needed, often in a way that needs minimal input from the user. *Relevance:* Semantic relevance of dynamically accessing information. By showing information only when needed an overload of information can be avoided and the user can concentrate better.

Display data. *Description:* Type of interaction that evokes visual effects on given data to give more insight about it. This includes playing or pausing a video and other cues such as odor. *Relevance:* Semantic relevance of enabling showcasing of information aspects. This can improve comprehension of the data.

Edit data. *Description:* Type of interaction that enables applying change to data. This includes adding, removing, or adjusting data elements. *Relevance:* Semantic relevance of enabling change the data set. Working on data is one reason why this could be included.

Edit selection. *Description:* Type of interaction that puts objects into the current selection or removes them from it. The selected data can then be interacted with. *Relevance:* Semantic relevance of generating active selection of objects. Induce effects only on the active elements.

Filter. *Description:* Type of interaction that filters data based on the selected criteria. In contrast to details-on-demand filtering can require more effort as the user has to specify aimed data through the query. *Relevance:* Semantic relevance of search and find by filter. This enables accessing relevant information by using the correct query.

Navigate data. *Description:* Type of interaction that modifies the virtual position of visualizations. This category is distinguished from ”Navigate view”due to semantic differences. In navigate data, the data is moved. *Relevance:* Semantic relevance of navigation. This allows the examination of data, evoking a semantic sensation similar to looking at objects held in hand.

Navigate view. *Description:* Type of interaction that modifies the viewpoint of the user. This category is distinguished from ”Navigate data”due to semantic differences. In navigate view, the view is moved. *Relevance:* Semantic relevance of navigation. This allows the exploration of data, evoking a semantic sensation similar to using for example a plane to explore environment.

Open. *Description:* Type of interaction that enables accessing or opening of a remote data or state. *Relevance:* Semantic relevance of accessing data or a state. This enables the user to open a none-active set of data that is saved somewhere. When used in combination with save it is possible to reopen an old state to work on.

Option control. *Description:* Type of interaction that accesses system options that are more complex and heavily rely on the field in which it is used. In classic HCI this is typically achieved with a UI such as a widget. Also included are inputs that execute specific controls or commands. For example in some application when used on Microsoft Windows this would include key combinations such as *ctrl* + *z* to undo an action. *Relevance:* Application specific functions that are too specific to fit another category.

Save. *Description:* Type of interaction that enables saving new data or a specific state for future use. *Relevance:* Semantic relevance of saving. Enables the user to save progress.

This concludes the 13 categories in which I classified interactions. For closer inspection of the relations between the categories, I created a preliminary concept diagram, included in Appendix A, in Figure A.1;

2.2 Results

In order to facilitate understanding the results, I split them into the ones containing results of hardware and software, and ones related to each category in the presented taxonomy.

2.2.1 Hardware and software

Many output hardware also incorporate input capabilities. For example, the Microsoft HoloLens device can track input gestures. Therefore, I decided to not mention output hardware when describing input hardware to avoid repetition. 34 studies that fit the criteria for this literature review used a Head-Mounted Display (HMD) as the output device. These devices corresponded mostly to the HTC Vive [CCD+17] (x13 times) and Oculus (6) Rift [MGHK15] for VR, and the Microsoft HoloLens [CCG19] (x12) for AR. However, there were also single case uses; namely a Samsung Odyssey HMD [HRD+19], Samsung Gear VR HMD [WWS20], Meta 1 HMD [NYW+16], and Google Cardboard [NWZ+16]. Some studies used the see-through property of AR HMD's as an advantage to combine them with monitors [FML+19; NYW+16; WBR+20]. In one case [PBE19], a HMD was combined with an olfactory display in order to incorporate scent, while another study [BHM+18] combined the HMD with a multi-touch table. Besides HMDs, some studies used mobile smart devices such as tablets or smartphones to realize AR. Large displays were always used in combination with other devices.

An interesting finding is the lack of immersive environments such as the Cave Automatic Virtual Environment (CAVE). Indeed, there was a single study [BBGS16] that fit the criteria and used such immersive environment. This may be due to fewer papers using such a setup to explore interaction to the level required in my literature review.

Input hardware are mostly controllers that complement HMD devices. Output devices that do not have a specific corresponding input device mostly involve the use of a computer mice, keyboards, controllers, and the Leap Motion[3] device. I also found an outlier study [KWFK19] that incorporates 3D pick ray, and another study [SJX+15] that uses a zSpace stylus pen[4].

The software used in the studies were mostly implemented with Unity3D[5]. However, in cases of web applications there were uses of JavaScript and WebGL. Toolkits especially designed to implement software for AR [JXK+19] or VR [SFX19] were also used. Table A.2, in Appendix A, presents detailed results of input and output hardware, and software. Missing information is denoted with a –. This is also the case if there is no additional Input Hardware (some Output Hardware can also receive input).

[3]https://www.ultraleap.com
[4]https://zspace.com
[5]https://unity.com

2.2.2 Interactions

The results are organized based on input devices, human actions required by input devices, and corresponding system reactions. Detailed results are presented in Appendix A. A reference column in the tables provides pointers to papers from which the data was extracted. Missing (or not applicable) information is displayed with a –. In the input column, a + sign denotes that human actions are achieved without a device or through a different category indicated in the human action column. In the following I present the findings.

Aggregate The results of aggregate interactions are presented in Table A.3. Half of the papers in which aggregation was found, implemented it using controllers. Others used devices capable of touch interactions. Also, there was a case in which a self-made device was used [CBC+20]. Results reveal that common approaches for aggregation are manual. Specifically, they include drawing using touch or bumping hands with controllers for clustering, grouping or merging. Volume slicing and general removal of objects (or part of objects) were done by removing an active intersection using either a controller, or as in one case, a self-made device [CBC+20]. In another case [ODZA18], the grab and throw metaphor was used to remove elements one by one. UI was used in only one case [BHM+18], enabling sorting.

Change Modes The results of change modes are presented in Table A.5. I found multiple input devices used to change modes but I noticed that human actions to perform this type of interaction are mostly based on metaphors of pushing a button or using a switch. When using the pushing a button metaphor, often I found the use of mice, keyboards, and controllers as input devices. Similarly, input devices involved when employing using a switch metaphor are *i*) controllers for swiping, *ii*) Leap Motion for gestures, and *iii*) various sensors to perceive tilt in the output device. The changing modes interactions involve a temporary as well as a persistent change. Both are implemented through the pushing a button metaphor by keeping the changed mode only active for as long as the button is pressed.

Data Scroll The results for *Data Scroll* are contained in Table A.7. I found that data scroll is often realized using swiping metaphors through touch compatible devices [SWKA19] or gestures [dJH+15]. However, there are also applications using a circular clock-like motion realized through the touch surface on a controller and the crown of a smartwatch [HBED18]. Only one case [BBGS16] involved pressing the button of a controller, and another [MSY+20] used a keyboard to implement this type of interaction.

Details-on-Demand The results for *Details-on-Demand* are contained in Table A.9. Sometimes users require details-on-demand to get contextual information of objects of interest. Often, this type of interaction was achieved by following users' gaze [MGHK15], head movements [MBD+18], and touching or picking up an object [ESMT16]. In these cases, actions that might fit the definition of other types of interactions, such as selection and data navigation, were used to determine interest in objects. There was only one case [FSN20] in which a controller was used triggering the system response with a touch input. This is interesting as it is the only case in which the details are revealed with neither a natural user interface nor in conjunction with a different type of interaction. Other applications used a voice command [MFM+19] to implement an on/off switch, and therefore, presumably not expecting it to be used much.

Display Data The results for *Display Data* are contained in Table A.11. There is not much of a pattern visible in the extracted table presumably due to the fact that this type of interaction defines a broad number of system reactions. This includes displaying data in forms such as playing a video [dJH+15] or execute a simulation [YLF+16]. Others visualize static elements such as visualizing 3D structures [FML+19]. Due to the lack of pattern it is interesting to consider the granularity of input data used that is needed for its specific purpose. In this context voice command [ERKL19], keyboard [WF09], and Leap Motion [dJH+15] were solely used for a discrete binary on / off functions. Sensors were used to trigger events such as using gestures to execute custom simulations [YLF+16]. A higher level of granularity for this interaction type was used in controllers using a custom made controller utilizing sliders [CBC+20] or a serial controller pressing a button and moving it [CBC+20]. Controllers were also used in the discrete sense using touch [CWT18]. In an outstanding case a controller button could be used to dispense odor [PBE19] indicating how diverse the ways data can be displayed can be.

Edit Data The results for *Edit Data* are contained in Table A.13. Depending on what kind of data was edited different actions were required. In one case [LSS17] a route should be manually set using a finger to draw it. In another case [MSY+20] program code should be typed using a keyboard. Adding new scorch rate samples was done on a smart device [WWS20]. Deletion was often implemented in an intuitive way by using, for instance, a crisscross touch motion on a smartwatch [HBED18], or a throwmetaphor [CCD+17]. Surprisingly, in other cases the complexity of editing data was rather low as it was implemented by pushing of a simple button. These results could suggest that IA/SV approaches have more focus on data analysis through exploration rather than producing, changing or removing data.

Edit Selection The results for *Edit Selection* are contained in Table A.15. Almost all studies in the literature review involve input devices used for editing selections, which is expected as selection is an interaction frequently required by most information visualizations. Often, selection is implemented by ray tracing, and used to choose an item followed by an action that confirms the choice, which enables a long range selection. The choosing part is achieved through either a pointing gesture, detecting gaze, or intersecting a virtual ray generated through a device with the item of interest. A selection is confirmed by pressing a button in the controller, or sometimes using a voice command. Some other times, selections are automatically confirmed when hold for a long enough time. In contrast to the long range selection some implementations used a short range selection, which requires either a touch action on the appropriate item or a collision of the device with the item. The former can be realized in AR either through a touch screen and subsequent ray tracing, or by directly touching the object using artificial intelligence to detect the chosen item.

Filter The findings for *Filter* are contained in Table A.17. Filtering is often included in other types of interactions such as filtering through selection, aggregation, or features in the UI. In one case [ESMT16], filtering was implemented in AR by locating real boxes side-by-side to enable multiple filters. In other cases filtering was implemented using a single button that had to be pressed [BBGS16] or air tap gestures [CCG19]. A much different approach used a virtual keyboard implemented with a Leap Motion device to enable dynamic queries [MGHK15].

Navigate Data The results for *Navigate Data* are presented in Table A.19. Navigation of data is a basic interaction for information visualization, and therefore, not surprisingly I found this interaction in multiple IA/SV papers. The majority of them follow a metaphor of natural movement as the human action required to address navigation. In some others, navigation is addressed using controllers inputs and system responses. For instance, pressing a button followed by moving the device [CCD+17] or hand [MSY+20] depending on whether the implementation includes a device or realizes it by hand gestures without a device. The exact type of gestures were not always specified in papers, but in cases they were specified, they were often implemented as follows. The one-handed way uses directional movement of the hand for an equivalent transfer of an object in space, rotation of the hand for a corresponding rotation of the object, a pinch gesture for scaling, or zooming, and a drag-and-drop metaphor to do so for an object. The two-handed transfer uses subsequent movement of both hands, for rotation both hands perform a clock-like circular motion, and zooming or scaling is done by increasing or decreasing the distance between both hands. In one case the scroll wheel of the mouse was used to allow more complex maneuvering [WBR+20]. The multi-touch table used a different approach enabling rotation and movement by pressing a button [BHM+18].

Navigate View The results for *Navigate View* are contained in Table A.21. The main input devices used for the navigation of the view are controllers that complement a corresponding HMD. In such cases, navigation is commonly achieved through teleportation [SWKA19]. This type of navigation is indicated in the table with the keyword Put in the System Reaction column. On top of a teleportation feature, I also found a few implementations in which the controller is used to gradually move in the scene [DCW+18]. In some other cases, input devices such as mouse or keyboard were used to enable navigation of the view. I observe that the lacking information on these devices is probably due to them being frequently used in a navigation context for which a more thorough description would be redundant. An interesting case [SJX+15] used a stylus pen combined with a 3D monitor as an alternative to using mouse or keyboard. The stylus pen works as the main tool avoiding users to have to change tools. In another interesting case [CDH16], in addition to using gaze in combination with a game pad for navigation, it is enabled to change viewpoints using vertical hand gestures.

Open Table A.23 contains the findings for *Open*. The input devices used for opening instances of data were either a smartwatch using touch gestures or a controller using a hold-and-release gesture. One paper [LNQ+19] describes the use of voice commands. I observe that there is small number of papers that describe open interactions. One reason to explain this could be that most IA/SV applications assume all data is already present and comply with an expected format. Meaning that any addition of more data is done before the immersive experience unfolds.

Option Control Findings for *Option Control* are presented in Table A.25. Option control defines a type of interaction that enables users to access some types of interaction in the first place. As such it is often realized by interacting with widgets [CCD+17] using a UI or visualizing the menu to be accessed [ESMT16]. However there are also other types of interaction that access meta-data changing parameters [LCPD19] or going beyond the experienced immersion to share visualizations with others [MFM+19]. Although in many cases devices such as controllers [ERKL19] were used there were also cases in which the interaction was realized using voice command [ERKL19] or natural gestures [JXK+19]. This finding is interesting as the naturalness of these ways to interact stands in stark contrast to this type of interaction, which is barely tangible due to its meta-properties.

Save Table A.27 contains the findings for *Save*. Among the input devices used for saving data, I found controllers for using a grab-and-pull gesture, and keyboards for pressing keys. One paper [BBGS16] describes the use of voice commands. I found a low number of papers that describe a save feature. One reason could be that most implementations automatically save data or in some cases do not store a state, and therefore, there is no need to save data. Reasons for the later could be that there is no change applied that would have to be saved. For example, instances when the purpose of the application is solely the exploration of data.

2.3 Discussion

In the following, I discuss findings collected from the results already presented.

As this investigation focus on a rather specific topic, the taxonomy I used to classify the data collected in the review of the literature of IA/SV presents a degree of novelty. Particularly, in some rare cases I found that the taxonomy is not as effective because these cases were not foreseen when defining the categories. For this reason I had modified the taxonomy once before. Still, I observe that interactions found across all IA/SV papers could be unambiguously classified. I identify one improvement, however, which I could not apply because required data was not collected. Specifically, the improvement involves changing the taxonomy in a way that would start the categorization process by first focusing on the outermost association layer. That is, when interacting with an IA/SV there is always an *action* in regards to its *effect*. However, the action does not have to be directly connected to the effect but can be connected through a transitive relation of associations. In the real-world there is no relevant association chain, but in HCI the interaction is mostly achieved through devices, and often, in addition to that with interfaces. By adding this concept into the proposed taxonomy it would result in changes that could give more reason to the constraints I made in order for only one category to apply for a specific interaction. Indeed, that is the specific case of the *option control* category, which then could be replaced by a category such as *interface control* if an action is achieved through a UI. The presented taxonomy already contains some of these association chains where input devices are omitted, and instead, categories used to enable actions are mentioned. Such cases can be further analyzed through results, in Appendix A, in Table A.3, Table A.9, Table A.11, Table A.13, Table A.15, Table A.17, and Table A.25.

Next, I analyze the ratio between the number of included papers and the number of initial papers found to study the proportion of papers that implement and describe interaction in sufficient detail. I found that only some IA/SV papers describe available interactions in detail. In particular, this case amounts to (38/187) 20.3% for IA, (10/167) 6% for SV, and (45/354) 12.7% for both results combined. I found many more IA/SV interaction details published in ACM DL than in IEEE Xplore. That is, I observe that (26/271) 9.6% of results from IEEE Xplore were included, and (19/83) 22.9% of results from ACM digital library were included. The reason for the divergence is at least to some degree the big number of none accurate results caused by using the standard search on IEEE Xplore while using the advanced search on ACM digital library.

When inspecting the results, I observed that in only few papers a keyboard was used as an input device [MSY+20], [WBR+20], [SJX+15]. In two of these cases [WBR+20], [SJX+15] a desktop setup was used in which mobility is already restricted, rendering the criteria of an easily wearable device unnecessary. However, a keyboard is an essential device to reliably and efficiently execute various controls and produce text. None of the other devices I found in the reviewed papers had these properties. In an effort to have an alternative to a restrictively mobile keyboard, in the following chapter I look at chorded wearable keyboards as an interaction means for IA/SV.

3 Application: Using chorded keyboards for IA/SV

In this chapter, I describe the design and implementation of an application to enable the use of chorded wearable keyboards for IA/SV. . Specifically, I chose the use of the Tap Strap 2 wearable keyboard as a possible device to replace the keyboard in cases when text should be generated in an immersive environment without inhibiting mobility. One such case could be the AVAR (Agile Visualizations in Immersive Augmented Reality)[1] SV application. AVAR is a toolkit that supports coding, and execution of code in the Pharo programming language[2]. For AVAR a keyboard is used as an input device [MSY+20] in order to enable users writing code displayed in an AR environment. With this as a use-case I aimed to create a mapping from the inputs of the Tap Strap 2 to outputs that can generate the syntax of Pharo. Besides the Tap Strap 2, I also considered other devices capable of replacing the less mobile keyboard. The Twiddler 3[3], for example, is a device that resembles a remote control to some extent. A kind of joystick attached to it enables control over a mouse cursor. In the end, I decided to use the Tap Strap 2, as in contrast to the Twiddler 3, for example, the user does not have to be aware of the positions of the buttons. In fact, this lack of spatial dependency and —for novices— visual dependency is more in accord with the key idea of my taxonomy.

The Tap Strap 2 is a wearable keyboard that can be worn on the hand like a glove. Instead of covering the whole hand, when worn correctly, it only clasps around the proximal phalanges with five flexible rings. A picture of the Tap Strap 2 is shown in Figure 3.1. On their website[4] the producers describe how the device works. In the following, I summarize important aspects of the device. Each of the Tap Strap 2 rings has a three-axis accelerometer sensor on it. In addition to that the thumb also contains a six-axis IMU (Inertial Measurement Unit), an optical mouse chip, haptic element, battery, and circuit board. When tapping on a surface with the fingers, the corresponding sensors detect it and send a signal to the circuit board Where the tap is processed. The signal is transmitted to the circuit board by means of conductors contained in the nylon braid that holds all the rings together. In the end, Bluetooth is used to send the resulting information to the device, the wearable keyboard is connected with. By default the Tap Strap 2 is set on tapping mode. That is, users are expected to tap with their fingers on surfaces. The combination of tapped fingers is detected by sensors and is taken as input. While resting the thumb on a surface, mouse mode is enabled. Left click and right click are realized by tapping the index finger or middle finger respectively and the cursor is moved by moving the thumb. In addition, an air gesture mode can be activated by holding the hand in a *hand shake* position. The air gesture mode enables emulating of a standard mouse; moving the cursor by moving the hand, clicking and scrolling by extending and flicking fingers. The number of possible taps using all fingers corresponds to $2^5 - 1 = 31$. This formula can be deduced by considering that the hand has five fingers, which can either be used for a particular tap or not. This

[1]`https://github.com/bsotomayor92/AVAR-unity`

[2]`https://pharo.org/`

[3]`https://twiddler.tekgear.com/`

[4]`https://www.tapwithus.com/how-tap-works/`

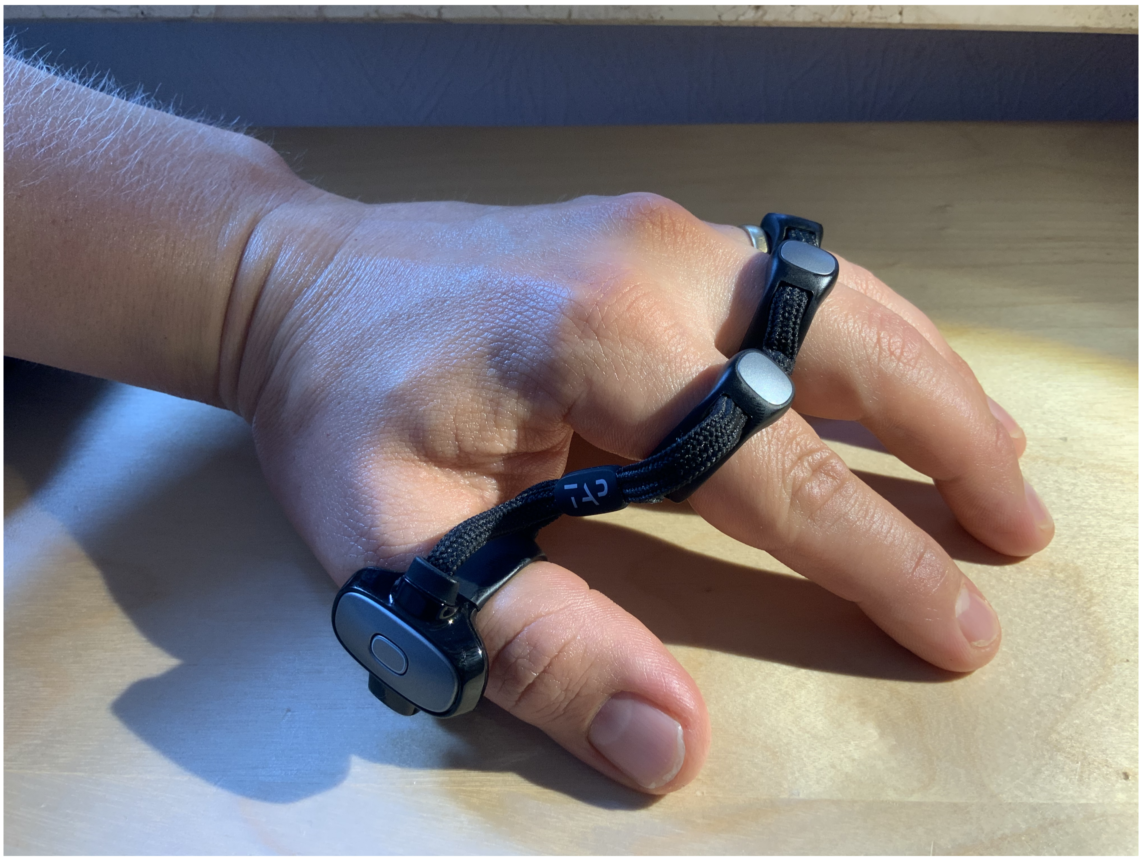


Figure 3.1: This is the Tap Strap 2 as worn on the hand.

concludes to 2^5. As each tap has to have at least one finger tapping, the case in which none is has to be subtracted. However, this number of possible taps is not enough to even allow all combinations of lower case letters of the English alphabet and all numbers. The Tap Strap 2 extends the number of possible inputs by containing different modes accessed with a specific tap combination. These modes consist of the default *Single Tap*, *Double Tap*, *Triple Tap*, *Shift*, and *Switch*. More details about these modes are shown in section 3.1. The design and discussion of the implementation is presented in the following.

3.1 Design

In subsection 3.1.1, I describe the design approaches I came up with and decide for one. Afterwards, in subsection 3.1.2, I state my approaches to create a fitting mapping for coding in Pharo.

3.1.1 Approaches and decision process

When deciding how I could implement a mapping that would enable Pharo code to be typed more easily I knew of two general approaches. One is using the custom mapping editor TapMapper Tool[5], which the Tap Strap 2 is shipped with. The other is using the Unity plugin for the Tap Strap 2[6] to implement my own mapping. In the following I assess both approaches and make a decision.

The Tap Strap 2 mapping editor enables users to choose what output is generated with particular inputs. However, it is limited in the number of inputs that can be accessed as well as how to access them. In the following, I list the accessibility and limitations of the mappings producible with the *Map Creator*.

`Single Tap`: *Access:* Active by default. When in *Switch* mode use designated tap to return to this mode. *Limitation:* Two tap combinations reserved for *Toggle Shift mode* and *Toggle Switch mode*. Another four tap combinations cannot be set to strings of text. Therefore at most 25 can be set with strings of text.

`Double Tap`: *Access:* Use specific tap two times in fast succession. *Limitation:* One tap combination disabled. Another three tap combinations reserved for *Take pic*, *Voice over*, and *Shift mode lock* respectively. When in *Single Tap* a specific tap combination has already been set to a string of text, that specific tap combination can no longer be used for *Double Tap*.

`Triple Tap`: *Access:* Use specific tap three times in fast succession. *Limitation:* Three tap combinations disabled by default. Another three tap combinations reserved for *Turn on*, *Turn off*, and *Show keyboard* respectively. When in either *Single Tap* or *Double Tap* a specific tap combination has already been set to a string of text, that specific tap combination can no longer be used for *Triple Tap*.

`Shift`: *Access:* In order to switch to this mode, use designated tap. After the tap is performed the mode changes back to *Single Tap*. *Limitation:* Two tap combinations reserved to toggle to *Shift* mode and *Switch* mode.

`Switch`: *Access:* In order to switch to this mode, use designated tap. *Limitation:* Changing to *Shift* mode disabled from this mode. Another two tap combinations reserved to *Toggle between mappings* and *Toggle Switch mode*.

The Tap Strap 2 mapping editor has the advantage that no further coding has to be done and the assigned inputs are directly produced by the wearable keyboard. This loose coupling enforces an excellent reusability. However, the above mentioned limitations make it less appropriate when trying to significantly expand on the default mapping or trying to create a mapping with many taps containing strings of text. Figure 3.2 shows the User interface for the Tap Mapper. The menu bar enables the user to switch between the different modes and multi taps.

The advantage of the Unity plugin for the Tap Strap 2 is that the inputs generated by the device are available for processing. Although loose coupling is limited in this fashion, the customizability is improved possibly avoiding the limitations of the Tap Mapper. Therefore, I decided to use the plugin to implement a mapping feature. The purpose of this mapping feature is to take the input from the

[5] `https://map.tapwithus.com/creator`

[6] `https://github.com/TapWithUs/tap-unity-plugin`

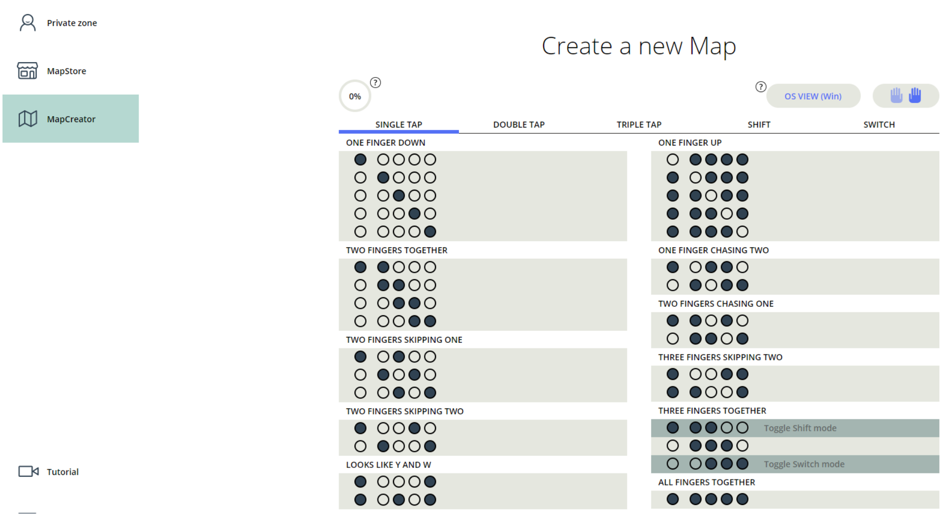


Figure 3.2: The official Tap Mapper for the Tap Strap 2

Tap Strap 2 and convert them into output text. My idea in doing so was taking the idea of using modes from the original mapping tool and expanding on it. To this end, I decided to implement a state machine based on the Unity plugin. Each mode can in this context be displayed by its own state which potentially has the full 31 number of possible inputs. Each input can have a possible input action, another state to transfer to or both. In order to make the mappings easily reusable, I decided to avoid hard-coding it but instead read the mapping from a file. This file should be in a format, that would make it easily readable and its content accessible. Because of the multi-dimensional nature of my design I decided on the JavaScript Object Notation (JSON) format. In this manner I only had to create a framework that could handle mappings stored with the JSON format. Doing so, users are enabled to specify their JSON file of choices and use it as a mapping as long as it conforms to the format that is required.

For the framework, I created classes as depicted in the class diagram in Figure 3.3. All of these classes are available on Github[7] including a brief guide on how to get them to work. Three of these classes C# object classes that reflect the structure I chose for the JSON files. The responsibility of their created instances is saving the information of the mapping. In the following, I describe these classes and their respective attributes.

`SingleInputMapping`: Is a class containing all necessary attributes that a specific tap has.

`mapID`: Is an integer and used as a unique identifier within a mode / state that corresponds to a specific input. Each input from the wearable keyboard is transformed into a specific number that can have a representation in a mode / state, in which case, the action specified therein is executed.

`transferMode`: Is a boolean that contains the information of whether or not this input transfers states.

`inputAction`: Is a string that contains the output produced by the input. If there is none it is empty.

[7] `https://github.com/NiklasMGaertner/TS2-WearableKeyboard-StateMachineMapper`

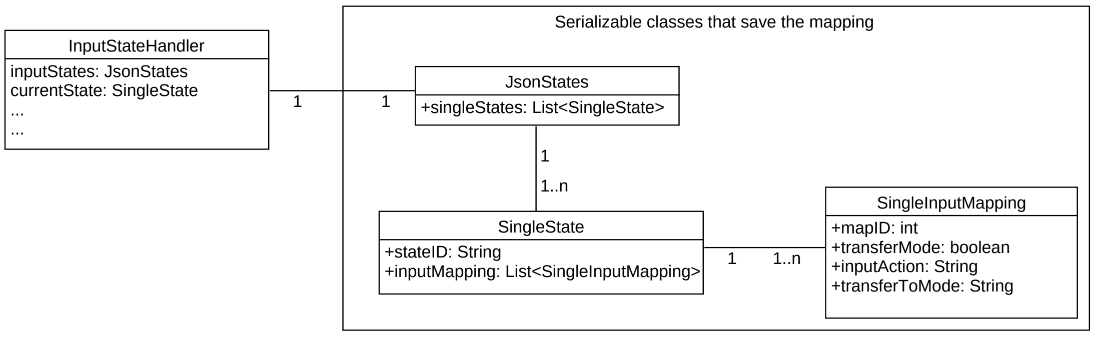


Figure 3.3: A class diagram depicting the implemented classes

`transferToMode`: Is a string that contains the mode / state that should be accessed next. This only happens if the `transferMode` is true.

`SingleState` Is a class containing all necessary attributes for a single state.

`stateID`: Is a string that contains the name of the mode / state. There should be one called `default` that is accessed first.

`inputMapping`: Is a list of `SingleInputMapping` that the mode / state has. Lastly there is a

`JsonStates`: Is a class containing only the list of `SingleState` that are present within the mapping. Through it all values can be accessed.

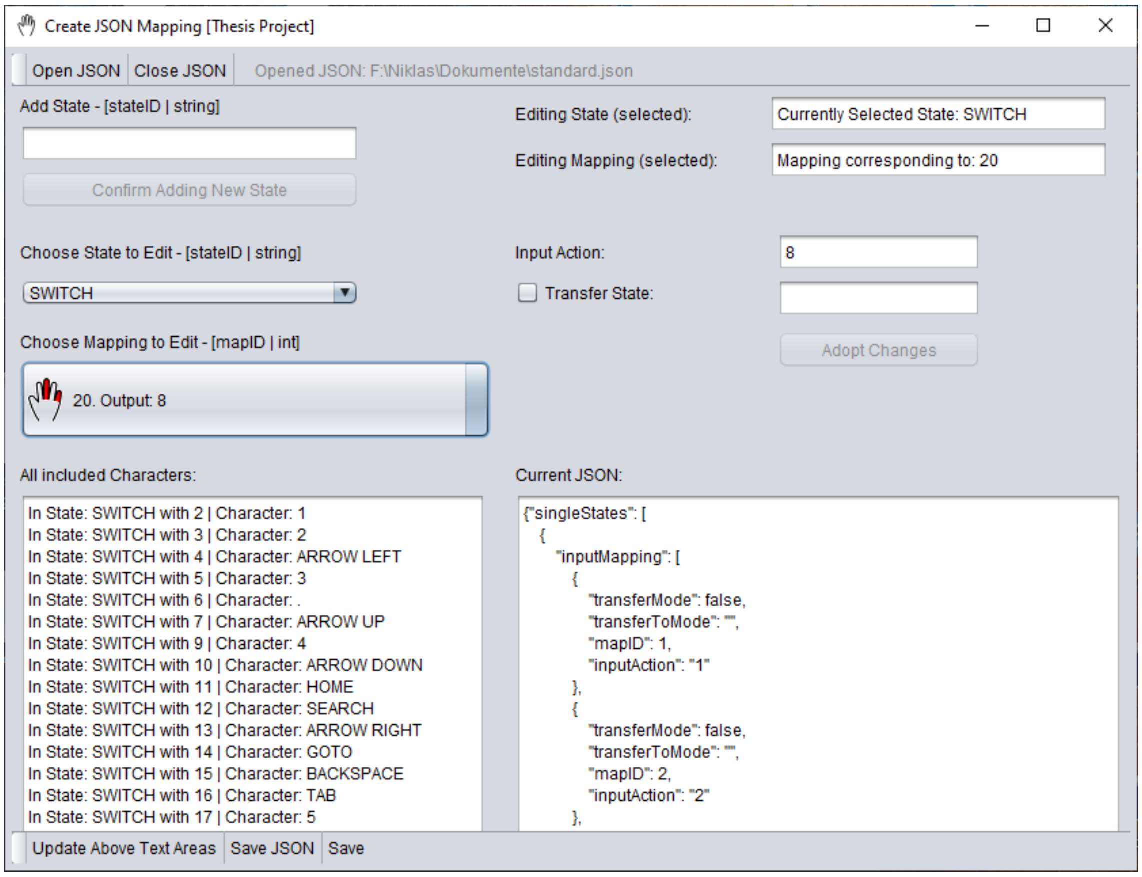
In the following a valid example file in JSON format for a mapping is shown.

```
{"singleStates": [
    {
        "inputMapping": [
            {
                "transferMode": true,
                "transferToMode": "A",
                "mapID": 1,
                "inputAction": ""
            },
            {
                "transferMode": false,
                "transferToMode": "",
                "mapID": 2,
                "inputAction": "b"
            }
        ],
        "stateID": "default"
    },
    {
        "inputMapping": [{
            "transferMode": false,
            "transferToMode": "default",
            "mapID": 1,
            "inputAction": "a"
        }],
        "stateID": "A"
    }
]}
```

The functionality of the `InputStateHandler` is processing all necessary information. On start it saves the mapping of the hard-coded file path into a `JsonStates` class containing instances of the `SingleState` class and the `SingleInputMapping` class. It also contains a state machine that transforms the inputs of the Tap Strap 2 to the unique `mapID` integer to determine the appropriate `inputAction`, `transferMode`, and `transferToMode` of the currently active mode. The current state is saved in the `currentState` variable and gets updated whenever the state changes. At the start the mode with the `stateID: default` is accessed. This class also contains the interface. To be specific it contains the return value of the state machine and can then be used for further processing. Alternatively it would have also been possible to simulate a button press as results of the state machine. However, this kind of design results in several security issues. Therefore I opted for the interface design.

3.1.2 Creation of mappings

I implemented a Java application to enable more easy creation of such mappings. The purpose of this application is avoiding syntax errors in the created mappings and make the creation more clearly arranged. Figure 3.4 displays the user interface I made for this purpose. In the upper menu bar there are buttons for opening or closing JSON files as well as a label showing the data path of the currently opened file. When opening, a pop-up window appears in which the file path can be chosen. Closing resets the label and set the used variables to null. A new mode can be added to the set by writing it down in the text field beneath the `Add State` label and confirmed by pressing the button. In order to edit a specific tap of a specific mode, it is necessary to first select the respective state in the drop-down list and then do the same for the drop-down list for the mapping. A clear view of the drop-down list for the mapping can be seen in Figure A.2. Fingers that are marked red indicate that they are tapped. The hand icons starting from 32 display the possible hand gestures of the Tap Strap 2 Unity plugin: `OneFingerUp`, `TwoFingersUp`, `OneFingerDown`, `TwoFingersDown`, `OneFingerLeft`, `TwoFingersLeft`, `OnefingerRight`, `TwoFingersRight`, `IndexToThumbTouch`, `MiddleToThumbTouch` The currently selected states can be observed in the non-editable text fields on the upper right corner of the widget. When both fields have an entry, the text field for input action and the transfer state checkbox can be edited. The text field to transfer states can be edited when its checkbox is set to true. In order to adopt changes the button beneath can be used. The bottom left text area shows a overview of all used characters and state transfer as well as the states from which they can be accessed. On the bottom right is a text area that displays how the resulting file in JSON format would look like. In the upper menu bar there are buttons for updating the text areas and saving. Although there was no case in my personal use of the application in which the updating of the text areas took particularly long, I still decided to use a button for it instead of updating every time something is changed. The reason for this is that in my personal use of the tool are not often necessary to look at. Saving the


Create JSON Mapping [Thesis Project]
Open JSON
Close JSON
Add State - [stateID | string]
Confirm Adding New State
Choose State to Edit - [stateID | string]
Choose Mapping to Edit - [mapID | int]
Editing State (selected):
Editing Mapping (selected):
Input Action:
Transfer State:
Adopt Changes
All included Characters:
Current JSON:
Update Above Text Areas
Save JSON
Save

Opened JSON: F:\Niklas\Dokumente\standard.json

Editing State (selected): Currently Selected State: SWITCH

Editing Mapping (selected): Mapping corresponding to: 20

Choose State to Edit: SWITCH

Choose Mapping to Edit: 20. Output: 8

Input Action: 8

All included Characters:

In State: SWITCH with 2 | Character: 1
In State: SWITCH with 3 | Character: 2
In State: SWITCH with 4 | Character: ARROW LEFT
In State: SWITCH with 5 | Character: 3
In State: SWITCH with 6 | Character: .
In State: SWITCH with 7 | Character: ARROW UP
In State: SWITCH with 9 | Character: 4
In State: SWITCH with 10 | Character: ARROW DOWN
In State: SWITCH with 11 | Character: HOME
In State: SWITCH with 12 | Character: SEARCH
In State: SWITCH with 13 | Character: ARROW RIGHT
In State: SWITCH with 14 | Character: GOTO
In State: SWITCH with 15 | Character: BACKSPACE
In State: SWITCH with 16 | Character: TAB
In State: SWITCH with 17 | Character: 5

Current JSON:

```
{"singleStates": [
  {
    "inputMapping": [
      {
        "transferMode": false,
        "transferToMode": "",
        "mapID": 1,
        "inputAction": "1"
      },
      {
        "transferMode": false,
        "transferToMode": "",
        "mapID": 2,
        "inputAction": "2"
      },
```

Figure 3.4: Screenshot displaying the user interface of my mapping tool

file can either be done by using `Save JSON` button in the lower menu bar or just the `Save` button next to it. The former opens a pop-up window to choose the file path, while the later uses the path seen in the label of the upper menu bar.

3.2 Mappings

To map the input from the wearable keyboard to an output string, there are several properties that I took into consideration.

Learnability: It should be easy for the user to get familiar with the mapping. This requires for the mapping to be made in a simple manner resulting in a better comprehension. It is also advantageous to take a mapping into consideration that the user is already accustomed to.

Usability: The mapping should enable the user to write text with low effort in a small amount of time. To achieve this, it is possible, for example, to make strings often used easier to produce in terms of tap effort and speed.

Completeness: It should be possible to create all text that is necessary with the mapping. That means that there should be a concatenation of one or more inputs by the wearable keyboard that produces an output string equivalent to the aimed text. Although this property is more of a base requirement it should still be mentioned as it has a big influence on the possibility of acquiring the other properties.

I made an effort to achieve the best trade-off between possible results in all these properties at the same. During the process of designing, I came up with various approaches that assign different priorities to these properties. Though, I think that the requirement for completeness is mandatory. Consequently, I defined the following three mappings.

Mapping A (priority: learnability) An initial idea to promote learnability was using a mapping familiar to the user. The Tap Strap 2 default mapping can be such a mapping as users can actively train to use the wearable keyboard using software for Android or iOS provided by the producers —TapAcademy[8].

Mapping B (priority: usability) When prioritizing efficiency, I first looked at the used syntax of the text that should be typed. The second step was to find appropriate strings of letters often used in conjunction in Pharo. Using such a string for a specific input simulates a macro that can make it possible for users to create text in a short amount of time with as few taps as possible. When creating the mapping for this approach, I considered the Pharo syntax[9] in order to determine the mapping. Certainly, macros alone are not enough to achieve completeness, and therefore, this mapping has to be supplemented with the possibility to produce single characters such as in mapping *A* or mapping *C*. To ensure that the mapping can be comprehended, I placed macros producing text of similar context in the same mode. A model of the initial idea for this mapping is depicted in Figure A.3. All additional strings of text I decided on can be found in Figure 3.5.

Mapping C (priority: learnability) A last approach for a mapping is to take the character frequency in Pharo into account. More frequently used characters should be more easily accessible and typed more easily. In order to identify the frequency of used characters the following Pharo code can be used.

```
chars := (Object witAllSubclasses flatCollect:[:c| methods flatCollect:[:m| m sourceCode]]) asBag.
oc := Dictionary new.
chars do:[:e | oc at: e put: (chars accurrencesOf: e)].
oc associations asSortedCollection: [:a :b| a value >= b value]
```

In the end I decided to implement Mapping B using my Java tool to create the necessary file in JSON format. As this tool does not have the multi-tap feature of the original mapping editor I had to put in some more modes and mode traverses in order to have the full set of characters included. The full mapping I ended up creating can be seen on Github[10].

[8] https://www.tapwithus.com/apps/

[9] https://ci.inria.fr/pharo-contribution/job/UpdatedPharoByExample/lastSuccessfulBuild/artifact/book-result/SyntaxNutshell/SyntaxNutshell.pdf

[10] https://github.com/NiklasMGaertner/TS2-WearableKeyboard-StateMachineMapper

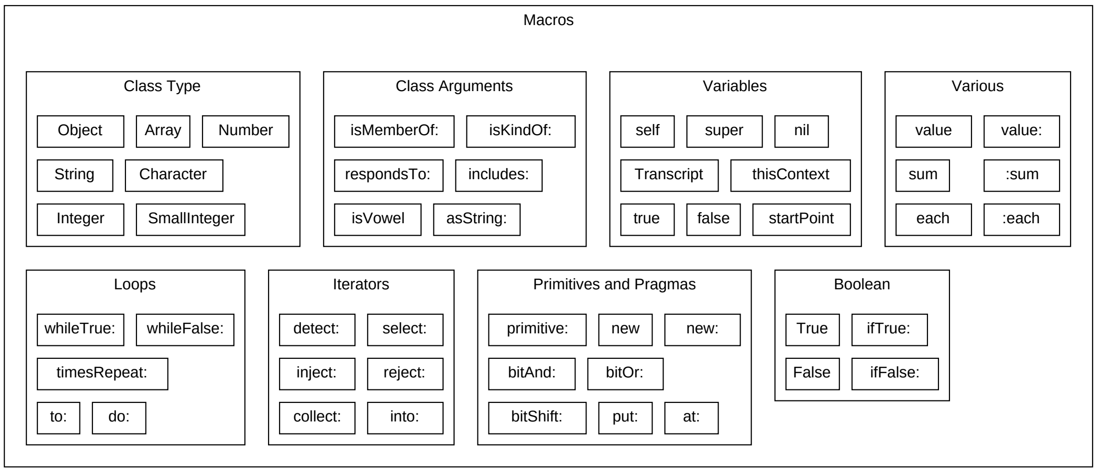


Figure 3.5: All additional strings of text (macros) I came up with for Mapping B

3.3 Discussion

The mapping approaches I designed are possibly not optimal. The mappings could benefit from Pharo expert feedback. For example, macros could be split into various groups in a way that macros that are frequently used in combination could be placed closer in terms of taps. The exact taps for macros might have differed as well to better reflect the frequency in which the text they contain appear. The aftermath of SARS-CoV-2 prevented me from conducting an expert evaluation to analyze properties of mappings. Originally it was also intended to apply my framework to AVAR to enable the use of these more complex mappings while using the Tap Strap 2. However problems when trying to install Roassal2[11] and Woden[12] prevented necessary testing in addition to the secure transfer of state machine return values.

When looking back on my mapping approaches, I think I could have put more thought into using the hand gesture capabilities of the Tap Strap 2 to make my mappings more effective. As gestures are accessed in a different way than taps, their use increases the time for authoring. However, they could be used for mode traversal. The reason why I decided to implement Mapping B is, that even though it prioritizes usability, its learnability is quite good due to TapAcademy. As for Mapping A, it is the standard mapping causing no further actions needed to use it. A problem with creating Mapping C is that I could not find a clear way to quantify the value of a specific tap. Of course, taps using only one finger are easier to perform, and therefore, might be more fit to a character used more often. Decisively mapping in this manner may not pay off, as for example, no matter if observing the character frequency in English or Pharo the most often used letter is *e*. I, therefore, conclude, that without further criteria regarding intuitive ease of a finger combination for tapping and more clear benefit of a character over another, Mapping C is not the best choice. There is probably a better mapping in this field of use for single characters than the standard mapping. The following criteria could be used to determine it, given that they can be evaluated:

[11] https://github.com/ObjectProfile/Roassal2
[12] https://github.com/ronsaldo/woden

- Character/Macro frequency (making them more easily accessible for usability)
- Character/Macro interplay (character/macro often used together placed so they can be accessed fast in concatenations)
- Character/Macro semantics (placing them so they can be accessed – similar modes)
- Macro decision process (deciding when to use macros, when not to, and when split one into smaller parts)

4 Conclusion

In this thesis, I explored interaction modalities for IA and SV. To do so, I first conducted a systematic literature review. Initially, I obtained 354 results from main digital libraries, from which I included 45 papers in my investigation that satisfied inclusion and exclusion criteria. The data extracted from papers was classified using a taxonomy to answer specific questions that I formulated. Specifically, the taxonomy not only classifies interactions based on their direct purpose but also on how natural they are to the user. I presented results using the taxonomy and discussed findings. One such finding was the scarce use of keyboards even though it posses unique strength in text input. One reason to explain it was that the low mobility of fixed keyboards is too restricting for IA and SV applications. Instead, mobile alternative keyboards could be investigated to support such applications. I then described the development of an application of a chorded wearable keyboard as an interaction device (which is not present amongst the analyzed papers) that could offer extended capabilities for complex text inputs such as when programming. Next, I elaborated on the particularities of using the Tap Strap 2 device to write code in Pharo. To do so, I extended the Unity plugin for the wearable keyboard to create a framework for a state machine capable of containing a higher quantity inputs. I conclude that using a wearable keyboard in this way could improve naturalness of the interaction as it *i*) has higher mobility than the classic keyboard, *ii*) does not need users to visually pinpoint its location, ultimately leading to *iii*) the users input action not being spatially constricted. However, the Tap Strap 2 also has disadvantages that the classic keyboard does not have. That is the lower number of distinct input options when disregarding the possibility of changing modes. Following the basic idea of my taxonomy the addition of more modes needed to enable the same number of input options can vastly increase the association chain, and therefore, decrease the naturalness. Still, I think that there are many cases in which chorded wearable keyboards are a good choice to complete immersive tasks. Therefore, I consider this type of input device enriching to the set of devices used in immersive contexts.

4.1 Limitations

The implications that can be made from the results of the literature review are limited to the selected sources (i. e., ACM DL and IEEE Xplore) when collecting papers. However, I selected such sources because I think that most IA/SV papers are indexed by them. Another limitation is imposed by the choice of not including papers based on forward and backward snowballing. However, I think this measure was need to keep the number of papers in a reasonable scope for the project.

The developed framework has the limitation that it can only process JSON files that have the required format described in subsection 3.1.1. Furthermore the data path to this file is not manually defined in order to avoid problems when using different devices. Instead the filename is hard-coded and has to be manually changed. The file is not easily accessible as it needs to be in the `.../User/AppData/LocalLow/<Company>/<ProjectName>` folder.

4.2 Future Work

In the introduction I attempted to spotlight the naturalness of user interactions, which impacted the construction of the taxonomy, and therefore, the classified data. However, the true value of natural interaction is unknown to me. In a brief search on IEEE Xplore with keywords such as *immersive non-immersive comparison* and *immersive comprehension* I looked at the first 20 results but did not find any article, journal or paper that investigated how a more natural interaction changes efficiency. Maybe a different set of keywords would reveal more results. In case a more thorough search like this does not return sufficient results this topic might be interesting for further research. A start for this could be creating a more refined taxonomy than mine that focuses on the natural properties of interaction. The basic idea for using such a taxonomy, would be aiming to extract the potential value natural interaction could have. My hypothesis for such value is that a more natural interaction requires less attention from the user. This could make it possible for users to concentrate more on their tasks and by that boosting their efficiency. Different devices but also natural user interfaces could then be compared.

Another idea for future research would be a user evaluation of using the Tap Strap 2 for coding in SV. To do so the framework specified here could be applied to AVAR. Of course, chorded keyboards could also be employed in other contexts, in which various functionality including text is needed and the mobility restricting keyboard is not appropriate. For this purpose other mappings could be created that do not have the weaknesses I described about mine. If the approach of using wearable keyboards in this manner proves to be useful, it might also be interesting to implement applications for learning to use developed mappings. Mappings could also be created for other domain-specific languages. Also an interface to use wearable keyboards could also be included in other IA/SV toolkits.

Bibliography

[BBGS16] N. Brunhart-Lupo, B. W. Bush, K. Gruchalla, S. Smith. "Simulation exploration through immersive parallel planes". In: *2016 Workshop on Immersive Analytics (IA)*. 2016, pp. 19–24 (cit. on pp. 21–24, 48–50, 52, 53, 55).

[BHM+18] S. Butscher, S. Hubenschmid, J. Müller, J. Fuchs, H. Reiterer. "Clusters, Trends, and Outliers: How Immersive Technologies Can Facilitate the Collaborative Analysis of Multidimensional Data". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–12. ISBN: 9781450356206. DOI: 10.1145/3173574.3173664. URL: https://doi.org/10.1145/3173574.3173664 (cit. on pp. 21, 22, 24, 48–53, 55).

[BM13] M. Brehmer, T. Munzner. "A Multi-Level Typology of Abstract Visualization Tasks". In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2376–2385 (cit. on p. 17).

[BMD18] W. Büschel, A. Mitschick, R. Dachselt. "Here and Now: Reality-Based Information Retrieval: Perspective Paper". In: *Proceedings of the 2018 Conference on Human Information Interaction Retrieval*. CHIIR '18. New Brunswick, NJ, USA: Association for Computing Machinery, 2018, pp. 171–180. ISBN: 9781450349253. DOI: 10.1145/3176349.3176384. URL: https://doi.org/10.1145/3176349.3176384 (cit. on pp. 48, 49, 52, 53).

[BTNB19] T. Bednarz, M. Tobia, H. Nguyen, D. Branchaud. "Immersive Analytics Using Augmented Reality for Computational Fluid Dynamics Simulations". In: *The 17th International Conference on Virtual-Reality Continuum and Its Applications in Industry*. VRCAI '19. Brisbane, QLD, Australia: Association for Computing Machinery, 2019. ISBN: 9781450370028. DOI: 10.1145/3359997.3365735. URL: https://doi.org/10.1145/3359997.3365735 (cit. on pp. 48–53, 55).

[CBC+20] M. Cordeil, B. Bach, A. Cunningham, B. Montoya, R. T. Smith, B. H. Thomas, T. Dwyer. "Embodied Axes: Tangible, Actuated Interaction for 3D Augmented Reality Data Spaces". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–12. ISBN: 9781450367080. DOI: 10.1145/3313831.3376613. URL: https://doi.org/10.1145/3313831.3376613 (cit. on pp. 22, 23, 48–51, 55).

[CCD+17] M. Cordeil, A. Cunningham, T. Dwyer, B. H. Thomas, K. Marriott. “ImAxes: Immersive Axes as Embodied Affordances for Interactive Multivariate Data Visualisation”. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST ’17. Québec City, QC, Canada: Association for Computing Machinery, 2017, pp. 71–83. ISBN: 9781450349819. DOI: 10.1145/3126594.3126613. URL: https://doi.org/10.1145/3126594.3126613 (cit. on pp. 21, 23, 24, 48, 49, 51–53, 55).

[CCG19] G. Caggianese, V. Colonnese, L. Gallo. “Situated Visualization in Augmented Reality: Exploring Information Seeking Strategies”. In: *2019 15th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*. 2019, pp. 390–395 (cit. on pp. 21, 23, 48, 49, 52, 53, 55).

[CDH16] M. Cordeil, T. Dwyer, C. Hurter. “Immersive Solutions for Future Air Traffic Control and Management”. In: *Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces*. ISS ’16 Companion. Niagara Falls, Ontario, Canada: Association for Computing Machinery, 2016, pp. 25–31. ISBN: 9781450345309. DOI: 10.1145/3009939.3009944. URL: https://doi.org/10.1145/3009939.3009944 (cit. on pp. 24, 48–51, 53, 54).

[CSW+20] Z. Chen, Y. Su, Y. Wang, Q. Wang, H. Qu, Y. Wu. “MARVisT: Authoring Glyph-Based Visualization in Mobile Augmented Reality”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.8 (2020), pp. 2645–2658 (cit. on pp. 48–51, 53, 55).

[CWT18] A. Cunningham, J. Walsh, B. Thomas. “Immersive Visualisation of Geo-Temporal Narratives in Law Enforcement”. In: *2018 International Symposium on Big Data Visual and Immersive Analytics (BDVA)*. 2018, pp. 1–8 (cit. on pp. 23, 48–51, 53).

[DCW+18] A. Drogemuller, A. Cunningham, J. Walsh, M. Cordeil, W. Ross, B. Thomas. “Evaluating Navigation Techniques for 3D Graph Visualizations in Virtual Reality”. In: *2018 International Symposium on Big Data Visual and Immersive Analytics (BDVA)*. 2018, pp. 1–10 (cit. on pp. 24, 48, 49, 52, 54).

[dJH+15] M. de Ridder, Y. Jung, R. Huang, J. Kim, D. D. Feng. “Exploration of Virtual and Augmented Reality for Visual Analytics and 3D Volume Rendering of Functional Magnetic Resonance Imaging (fMRI) Data”. In: *2015 Big Data Visual Analytics (BDVA)*. 2015, pp. 1–8 (cit. on pp. 22, 23, 48–51, 53).

[ERKL19] U. Engelke, C. Rogers, J. Klump, I. Lau. “HypAR: Situated Mineralogy Exploration in Augmented Reality”. In: *The 17th International Conference on Virtual-Reality Continuum and Its Applications in Industry*. VRCAI ’19. Brisbane, QLD, Australia: Association for Computing Machinery, 2019. ISBN: 9781450370028. DOI: 10.1145/3359997.3365715. URL: https://doi.org/10.1145/3359997.3365715 (cit. on pp. 23, 24, 48, 49, 51, 52, 55).

[ESMT16] N. A. M. ElSayed, R. T. Smith, K. Marriott, B. H. Thomas. “Blended UI Controls for Situated Analytics”. In: *2016 Big Data Visual Analytics (BDVA)*. 2016, pp. 1–8 (cit. on pp. 22–24, 48–53, 55).

[ETM+15] N. ElSayed, B. Thomas, K. Marriott, J. Piantadosi, R. Smith. “Situated Analytics”. In: *2015 Big Data Visual Analytics (BDVA)*. 2015, pp. 1–8 (cit. on pp. 48, 49, 52, 53, 55).

[FML+19] W. Fulmer, T. Mahmood, Z. Li, S. Zhang, J. Huang, A. Lu. “ImWeb: Cross-Platform Immersive Web Browsing for Online 3D Neuron Database Exploration”. In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. IUI ’19. Marina del Ray, California: Association for Computing Machinery, 2019, pp. 367–378. ISBN: 9781450362726. DOI: 10.1145/3301275.3302319. URL: https://doi.org/10.1145/3301275.3302319 (cit. on pp. 21, 23, 48–53).

[FSN20] J. A. W. Filho, W. Stuerzlinger, L. Nedel. “Evaluating an Immersive Space-Time Cube Geovisualization for Intuitive Trajectory Data Exploration”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 514–524 (cit. on pp. 22, 48, 49, 51–53).

[HBED18] T. Horak, S. K. Badam, N. Elmqvist, R. Dachselt. “When David Meets Goliath: Combining Smartwatches with a Large Vertical Display for Visual Data Exploration”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI ’18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–13. ISBN: 9781450356206. DOI: 10.1145/3173574.3173593. URL: https://doi.org/10.1145/3173574.3173593 (cit. on pp. 22, 23, 48–55).

[HRD+19] C. Hurter, N. H. Riche, S. M. Drucker, M. Cordeil, R. Alligier, R. Vuillemot. “FiberClay: Sculpting Three Dimensional Trajectories to Reveal Structural Insights”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 704–714 (cit. on pp. 21, 48, 49, 52–55).

[IDJW19] A. Ivanov, K. Danyluk, C. Jacob, W. Willett. “A Walk Among the Data”. In: *IEEE Computer Graphics and Applications* 39.3 (2019), pp. 19–28 (cit. on pp. 48–51, 53).

[JKJ+19] S. Jaeger, K. Klein, L. Joos, J. Zagermann, M. de Ridder, J. Kim, J. Yang, U. Pfeil, H. Reiterer, F. Schreiber. “Challenges for Brain Data Analysis in VR Environments”. In: *2019 IEEE Pacific Visualization Symposium (PacificVis)*. 2019, pp. 42–46 (cit. on pp. 48, 49, 52, 53, 55).

[JXK+19] A. Jing, C. Xiang, S. Kim, M. Billinghurst, A. Quigley. “SnapChart: An Augmented Reality Analytics Toolkit to Enhance Interactivity in a Collaborative Environment”. In: *The 17th International Conference on Virtual-Reality Continuum and Its Applications in Industry*. VRCAI ’19. Brisbane, QLD, Australia: Association for Computing Machinery, 2019. ISBN: 9781450370028. DOI: 10.1145/3359997.3365725. URL: https://doi.org/10.1145/3359997.3365725 (cit. on pp. 21, 24, 48, 49, 52, 53, 55).

[KB13] B. Kitchenham, P. Brereton. “A systematic review of systematic review process research in software engineering”. In: *Information and Software Technology* 55 (Dec. 2013), pp. 2049–2075. DOI: 10.1016/j.infsof.2013.07.010 (cit. on p. 15).

[KFC+19] A. Knote, S. C. Fischer, S. Cussat-Blanc, F. Niebling, D. Bernard, F. Cogoni, S. von Mammen. “Immersive Analysis of 3D Multi-cellular In-Vitro and In-Silico Cell Cultures”. In: *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. 2019, pp. 82–827 (cit. on pp. 48–50, 53, 55).

[KWFK19] A. Kunert, T. Weissker, B. Froehlich, A. Kulik. “Multi-Window 3D Interaction for Collaborative Virtual Reality”. In: *IEEE Transactions on Visualization and Computer Graphics* (2019), pp. 1–1 (cit. on pp. 21, 48, 49, 51–53).

[LaV20] LaValle. *VIRTUAL REALITY*. 2020. URL: http://lavalle.pl/vr/ (cit. on p. 13).

[LCPD19] B. Lee, M. Cordeil, A. Prouzeau, T. Dwyer. "FIESTA: A Free Roaming Collaborative Immersive Analytics System". In: *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*. ISS '19. Daejeon, Republic of Korea: Association for Computing Machinery, 2019, pp. 335–338. ISBN: 9781450368919. DOI: 10.1145/3343055.3360746. URL: https://doi.org/10.1145/3343055.3360746 (cit. on pp. 24, 48, 49, 52–55).

[LNQ+19] C. W. Lau, Q. V. Nguyen, Z. Qu, S. Simoff, D. Catchpoole. "Immersive Intelligence Genomic Data Visualisation". In: *Proceedings of the Australasian Computer Science Week Multiconference*. ACSW 2019. Sydney, NSW, Australia: Association for Computing Machinery, 2019. ISBN: 9781450366038. DOI: 10.1145/3290688.3290722. URL: https://doi.org/10.1145/3290688.3290722 (cit. on pp. 24, 48–54).

[LSS17] N. Li, E. Sharlin, M. C. Sousa. "Duopography: Using Back-of-Device Multi-Touch Input to Manipulate Spatial Data on Mobile Tangible Interactive Topography". In: *SIGGRAPH Asia 2017 Mobile Graphics Interactive Applications*. SA '17. Bangkok, Thailand: Association for Computing Machinery, 2017. ISBN: 9781450354103. DOI: 10.1145/3132787.3139197. URL: https://doi.org/10.1145/3132787.3139197 (cit. on pp. 23, 48–53).

[MBD+18] T. Mahmood, E. Butler, N. Davis, J. Huang, A. Lu. "Building Multiple Coordinated Spaces for Effective Immersive Analytics through Distributed Cognition". In: *2018 International Symposium on Big Data Visual and Immersive Analytics (BDVA)*. 2018, pp. 1–11 (cit. on pp. 22, 48–53).

[MBN18] L. Merino, A. Bergel, O. Nierstrasz. "Overcoming Issues of 3D Software Visualization through Immersive Augmented Reality". In: *2018 IEEE Working Conference on Software Visualization (VISSOFT)*. 2018, pp. 54–64 (cit. on p. 13).

[MFM+19] T. Mahmood, W. Fulmer, N. Mungoli, J. Huang, A. Lu. "Improving Information Sharing and Collaborative Analysis for Remote GeoSpatial Visualization Using Mixed Reality". In: *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 2019, pp. 236–247 (cit. on pp. 22, 24, 48–53, 55).

[MGHK15] A. Moran, V. Gadepally, M. Hubbell, J. Kepner. "Improving Big Data visual analytics with interactive virtual reality". In: *2015 IEEE High Performance Extreme Computing Conference (HPEC)*. 2015, pp. 1–6 (cit. on pp. 21–23, 48, 49, 51, 53).

[MGO+19] S. Mirhosseini, I. Gutenko, S. Ojal, J. Marino, A. Kaufman. "Immersive Virtual Colonoscopy". In: *IEEE Transactions on Visualization and Computer Graphics* 25.5 (2019), pp. 2011–2021 (cit. on pp. 48–50, 52–54).

[MHB+19] L. Merino, M. Hess, A. Bergel, O. Nierstrasz, D. Weiskopf. "PerfVis: Pervasive Visualization in Immersive Augmented Reality for Performance Awareness". In: *Companion of the 2019 ACM/SPEC International Conference on Performance Engineering*. ICPE '19. Mumbai, India: Association for Computing Machinery, 2019, pp. 13–16. ISBN: 9781450362863. DOI: 10.1145/3302541.3313104. URL: https://doi.org/10.1145/3302541.3313104 (cit. on p. 13).

[MSY+20] L. Merino, B. Sotomayor-Gómez, X. Yu, R. Salgado, A. Bergel, M. Sedlmair, D. Weiskopf. "Toward Agile Situated Visualization: An Exploratory User Study". In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing*

Systems. CHI EA '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–7. ISBN: 9781450368193. DOI: 10.1145/3334480.3383017. URL: https://doi.org/10.1145/3334480.3383017 (cit. on pp. 22–25, 27, 48–53, 55).

[NB16] M. Nabiyouni, D. A. Bowman. "A Taxonomy for Designing Walking-Based Locomotion Techniques for Virtual Reality". In: *Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces*. ISS '16 Companion. Niagara Falls, Ontario, Canada: Association for Computing Machinery, 2016, pp. 115–121. ISBN: 9781450345309. DOI: 10.1145/3009939.3010076. URL: https://doi.org/10.1145/3009939.3010076 (cit. on p. 18).

[NWZ+16] H. T. Nim, M. Wang, Y. Zhu, B. Sommer, F. Schreiber, S. E. Boyd, S. J. Wang. "Communicating the Effect of Human Behaviour on the Great Barrier Reef via Mixed Reality Visualisation". In: *2016 Big Data Visual Analytics (BDVA)*. 2016, pp. 1–6 (cit. on pp. 21, 48, 49, 53).

[NYW+16] K. Nagao, Y. Ye, C. Wang, I. Fujishiro, K. Ma. "Enabling interactive scientific data visualization and analysis with see-through hmds and a large tiled display". In: *2016 Workshop on Immersive Analytics (IA)*. 2016, pp. 1–6 (cit. on pp. 21, 48, 49, 53, 55).

[ODZA18] T. Onorati, P. Díaz, T. Zarraonandia, I. Aedo. "The Immersive Bubble Chart: A Semantic and Virtual Reality Visualization for Big Data". In: *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*. UIST '18 Adjunct. Berlin, Germany: Association for Computing Machinery, 2018, pp. 176–178. ISBN: 9781450359498. DOI: 10.1145/3266037.3271642. URL: https://doi.org/10.1145/3266037.3271642 (cit. on pp. 22, 48–50, 53, 55).

[PBE19] B. Patnaik, A. Batch, N. Elmqvist. "Information Olfactation: Harnessing Scent to Convey Data". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 726–736 (cit. on pp. 21, 23, 48, 49, 51).

[SFX19] B. Sun, A. Fritz, W. Xu. "An Immersive Visual Analytics Platform for Multidimensional Dataset". In: *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*. 2019, pp. 24–29 (cit. on pp. 21, 48, 49, 51, 53, 54).

[SH16] D. Schmalstieg, T. Höllerer. *Augmented Reality: Principles and Practice*. Addison-Wesley usability and HCI series. Addison-Wesley, 2016. ISBN: 9780321883575. URL: https://books.google.de/books?id=f5GAMAEACAAJ (cit. on p. 13).

[Shn96] B. Shneiderman. "The eyes have it: a task by data type taxonomy for information visualizations". In: *Proceedings 1996 IEEE Symposium on Visual Languages*. 1996, pp. 336–343 (cit. on p. 18).

[SJX+15] B. Sommer, S. Jia Wang, L. Xu, M. Chen, F. Schreiber. "Hybrid-Dimensional Visualization and Interaction - Integrating 2D and 3D Visualization with Semi-Immersive Navigation Techniques". In: *2015 Big Data Visual Analytics (BDVA)*. 2015, pp. 1–8 (cit. on pp. 21, 24, 25, 48–50, 52–54).

[SWK+16] M. Simpson, J. O. Wallgrün, A. Klippel, L. Yang, G. Garner, K. Keller, D. Oprean, S. Bansal. "Immersive Analytics for Multi-Objective Dynamic Integrated Climate-Economy (DICE) Models". In: *Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces*. ISS '16 Companion. Niagara Falls, Ontario, Canada:

Association for Computing Machinery, 2016, pp. 99–105. ISBN: 9781450345309. DOI: 10.1145/3009939.3009955. URL: https://doi.org/10.1145/3009939.3009955 (cit. on pp. 48, 49, 53, 54).

[SWKA19] J. Sorger, M. Waldner, W. Knecht, A. Arleo. "Immersive Analytics of Large Dynamic Networks via Overview and Detail Navigation". In: *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. 2019, pp. 144–1447 (cit. on pp. 22, 24, 48–54).

[WBR+20] X. Wang, L. Besançon, D. Rousseau, M. Sereno, M. Ammi, T. Isenberg. "Towards an Understanding of Augmented Reality Extensions for Existing 3D Data Analysis Tools". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376657. URL: https://doi.org/10.1145/3313831.3376657 (cit. on pp. 21, 24, 25, 48–50, 53).

[WCST18] J. Walsh, A. Cunningham, R. Smith, B. Thomas. "Tangible Braille Plot: Tangibly Exploring Geo-Temporal Data in Virtual Reality". In: *2018 International Symposium on Big Data Visual and Immersive Analytics (BDVA)*. 2018, pp. 1–6 (cit. on pp. 48–51, 55).

[WF09] S. White, S. Feiner. "SiteLens: Situated Visualization Techniques for Urban Site Visits". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: Association for Computing Machinery, 2009, pp. 1117–1120. ISBN: 9781605582467. DOI: 10.1145/1518701.1518871. URL: https://doi.org/10.1145/1518701.1518871 (cit. on pp. 23, 48–51, 53, 55).

[WWS20] M. Whitlock, K. Wu, D. A. Szafir. "Designing for Mobile and Immersive Visual Analytics in the Field". In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 503–513 (cit. on pp. 21, 23, 48, 49, 52, 53).

[YKSJ07] J. S. Yi, Y. a. Kang, J. Stasko, J. Jacko. "Toward a Deeper Understanding of the Role of Interaction in Information Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (Nov. 2007), pp. 1224–1231. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.70515. URL: https://doi.org/10.1109/TVCG.2007.70515 (cit. on p. 17).

[YLF+16] D. Yim, G. N. Loison, F. H. Fard, E. Chan, A. McAllister, F. Maurer. "Gesture-Driven Interactions on a Virtual Hologram in Mixed Reality". In: *Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces*. ISS '16 Companion. Niagara Falls, Ontario, Canada: Association for Computing Machinery, 2016, pp. 55–61. ISBN: 9781450345309. DOI: 10.1145/3009939.3009948. URL: https://doi.org/10.1145/3009939.3009948 (cit. on pp. 23, 48, 49, 51–53).

All links were last followed on April 27, 2020.

A Appendix

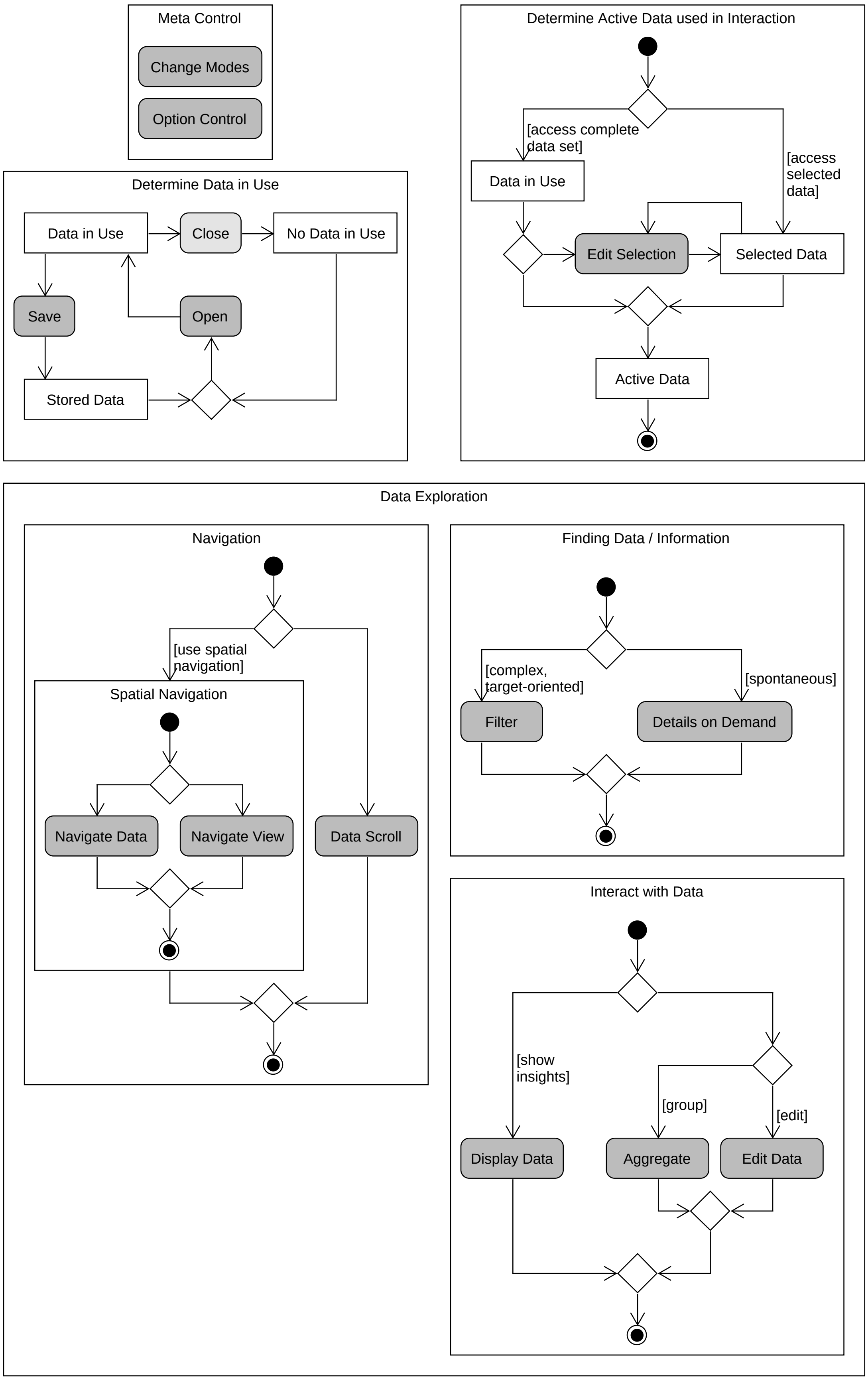


Figure A.1: My concept for the relations between the different categories of my taxonomy.

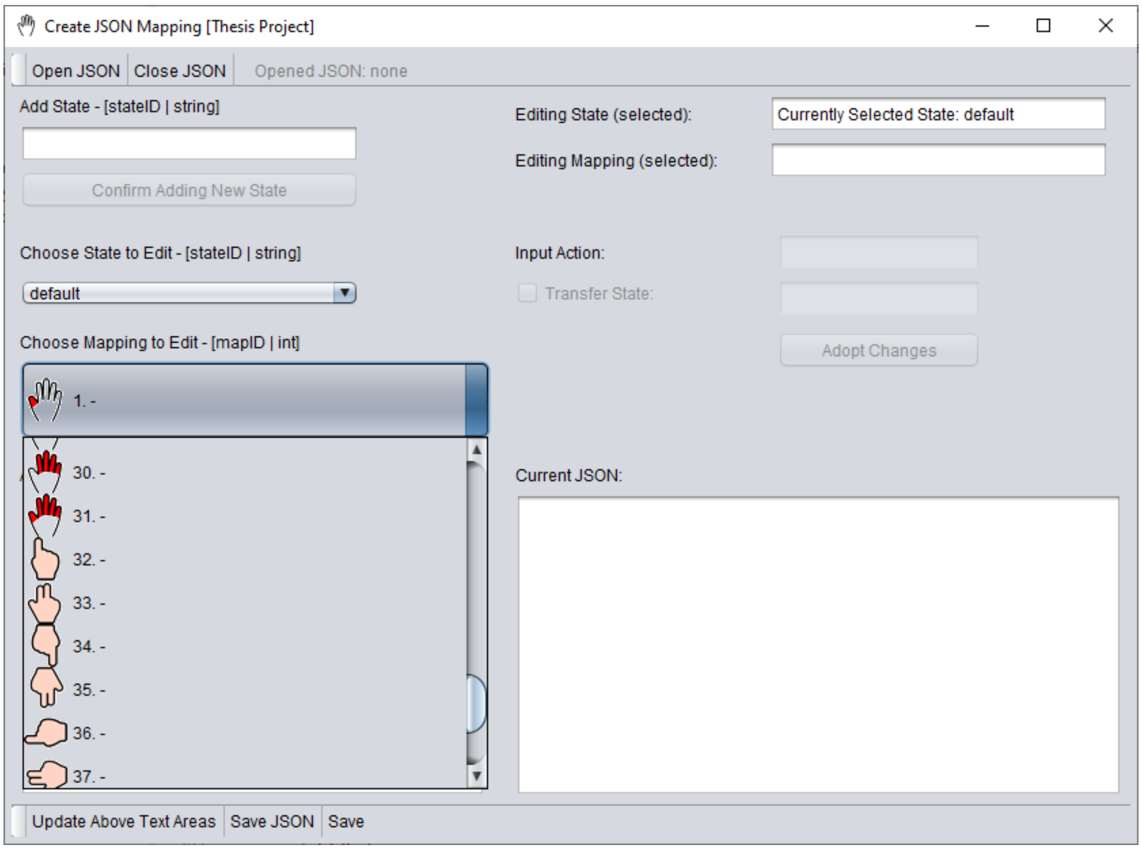

Create JSON Mapping [Thesis Project]
Open JSON
Close JSON
Opened JSON: none
Add State - [stateID | string]
Confirm Adding New State
Choose State to Edit - [stateID | string]
default
Choose Mapping to Edit - [mapID | int]
1. -
30. -
31. -
32. -
33. -
34. -
35. -
36. -
37. -
Editing State (selected):
Currently Selected State: default
Editing Mapping (selected):
Input Action:
Transfer State:
Adopt Changes
Current JSON:
Update Above Text Areas
Save JSON
Save

Figure A.2: Screenshot that shows the drop down menu user interface of my mapping tool

Table A.1: Titles and references of studies that are included in the literature review

Title	**References**
A Walk Among the Data	[IDJW19]
An Immersive Visual Analytics Platform for Multidimensional Dataset	[SFX19]
Blended UI Controls for Situated Analytics	[ESMT16]
Building Multiple Coordinated Spaces for Effective Immersive Analytics through Distributed Cognition	[MBD+18]
Challenges for Brain Data Analysis in VR Environments	[JKJ+19]
Clusters, Trends, and Outliers: How Immersive Technologies Can Facilitate the Collaborative Analysis of Multidimensional Data	[BHM+18]
Communicating the Effect of Human Behaviour on the Great Barrier Reef via Mixed Reality Visualisation	[NWZ+16]
Designing for Mobile and Immersive Visual Analytics in the Field	[WWS20]
Duopography: Using Back-of-Device Multi-Touch Input to Manipulate Spatial Data on Mobile Tangible Interactive Topography	[LSS17]
Embodied Axes: Tangible, Actuated Interaction for 3D Augmented Reality Data Spaces	[CBC+20]
Enabling interactive scientific data visualization and analysis with see-through hmds and a large tiled display	[NYW+16]
Evaluating an Immersive Space-Time Cube Geovisualization for Intuitive Trajectory Data Exploration	[FSN20]
Evaluating Navigation Techniques for 3D Graph Visualizations in Virtual Reality	[DCW+18]
Exploration of Virtual and Augmented Reality for Visual Analytics and 3D Volume Rendering of Functional Magnetic Resonance Imaging (fMRI) Data	[dJH+15]
FiberClay: Sculpting Three Dimensional Trajectories to Reveal Structural Insights	[HRD+19]
FIESTA: A Free Roaming Collaborative Immersive Analytics System	[LCPD19]
Gesture-Driven Interactions on a Virtual Hologram in Mixed Reality	[YLF+16]
Here and Now: Reality-Based Information Retrieval: Perspective Paper	[BMD18]
Hybrid-Dimensional Visualization and Interaction - Integrating 2D and 3D Visualization with Semi-Immersive Navigation Techniques	[SJX+15]
HypAR: Situated Mineralogy Exploration in Augmented Reality	[ERKL19]
ImAxes: Immersive Axes as Embodied Affordances for Interactive Multivariate Data Visualisation	[CCD+17]
Immersive Analysis of 3D Multi-cellular In-Vitro and In-Silico Cell Cultures	[KFC+19]
Immersive Analytics for Multi-Objective Dynamic Integrated Climate-Economy (DICE) Models	[SWK+16]
Immersive Analytics of Large Dynamic Networks via Overview and Detail Navigation	[SWKA19]
Immersive Analytics Using Augmented Reality for Computational Fluid Dynamics Simulations	[BTNB19]
Immersive Intelligence Genomic Data Visualisation	[LNQ+19]
Immersive Solutions for Future Air Traffic Control and Management	[CDH16]
Immersive Virtual Colonoscopy	[MGO+19]
Immersive Visualisation of Geo-Temporal Narratives in Law Enforcement	[CWT18]
Improving Big Data visual analytics with interactive virtual reality	[MGHK15]
Improving Information Sharing and Collaborative Analysis for Remote GeoSpatial Visualization Using Mixed Reality	[MFM+19]
ImWeb: Cross-Platform Immersive Web Browsing for Online 3D Neuron Database Exploration	[FML+19]
Information Olfactation: Harnessing Scent to Convey Data	[PBE19]
MARVisT: Authoring Glyph-based Visualization in Mobile Augmented Reality	[CSW+20]
Multi-Window 3D Interaction for Collaborative Virtual Reality	[KWFK19]
Simulation exploration through immersive parallel planes	[BBGS16]
SiteLens: Situated Visualization Techniques for Urban Site Visits	[WF09]
Situated Analytics	[ETM+15]
Situated Visualization in Augmented Reality: Exploring Information Seeking Strategies	[CCG19]
SnapChart: An Augmented Reality Analytics Toolkit to Enhance Interactivity in a Collaborative Environment	[JXK+19]
Tangible Braille Plot: Tangibly Exploring Geo-Temporal Data in Virtual Reality	[WCST18]
The Immersive Bubble Chart: A Semantic and Virtual Reality Visualization for Big Data	[ODZA18]
Toward Agile Situated Visualization: An Exploratory User Study	[MSY+20]
Towards an Understanding of Augmented Reality Extensions for Existing 3D Data Analysis Tools	[WBR+20]
When David Meets Goliath: Combining Smartwatches with a Large Vertical Display for Visual Data Exploration	[HBED18]

Table A.2: Results for: Extracted Hardware and Software information

Output Hardware	**Input Hardware**	**Software Information**	**Refs**
Epson BT-200, iPad Air	flat multi-touch surface, Leap Motion, irregular topographic surface	–	[LSS17]
Google Cardboard + smartphone	joystick / Leap Motion / computer mouse	Vuforia AR SDK and HTML, JavaScript	[NWZ+16]
HMD (HTC Vive)	6 DOF tracked hand controller	IATK, Unity	[CCD+17]
	HTC Vive controller	IATK toolkit	[CBC+20]
		JavaScript	[SWKA19]
		Unity	[MGO+19], [IDJW19], [SWK+16]
	HTC Vive controllers	Unity, VRTK package	[SFX19]
	HTC Vive markers; self-made controller	–	[WCST18]
	HTC Vive touch controllers	–	[CWT18]
	HTC Vive wand controllers	Unity	[KFC+19]
HMD (HTC Vive), multi-touch table	–	Unity	[BHM+18]
HMD (HTC Vive), olfactory display	–	Unity	[PBE19]
HMD (HTC Vive Pro)	HTC Vive controller	Unity	[LCPD19]
	HTC Vive touch controllers	Unity	[DCW+18]
HMD (Meta 1), Large Display	RGB Camera and depth sensor	–	[NYW+16]
HMD (Microsoft HoloLens)	–	–	[CCG19], [BMD18], [MBD+18]
		Unity	[LNQ+19], [BTNB19]
		Unity, C#	[ERKL19]
		Unity, HoloToolkit	[MFM+19]
	Apple Magic Bluetooth Keyboard	Pharo, Roassal2, Unity	[MSY+20]
	Microsoft Kinect 2	–	[YLF+16]
HMD (Microsoft Hololens) + Android Mobile Devices / HMD (Samsung Gear VR)	–	Unity, MySQL	[WWS20]
HMD (Microsoft HoloLens) + large display	–	Unity	[FML+19]
HMD (Microsoft HoloLens), (PC) Monitor	mouse + keyboard	Unity, C#, .NET	[WBR+20]
HMD (Oculus Rift)	Leap Motion	Unity	[MGHK15]
		WebGL	[dJH+15]
	Leap Motion, game pad	Unity	[CDH16]
	Oculus Rift Touch controllers	Unity	[ODZA18]
		Unity, R, JS D3	[JKJ+19]
HMD (Oculus Rift CV1)	Oculus Touch hand controllers	Unity	[FSN20]
HMD (Samsung Odyssey)	hand-held controllers with accelerometers for tracking	C#, DirectX	[HRD+19]
IE at Energy Systems Facility at National Renewable Energy Laboratory	Joystick controller Integration (and virtual tool)	C++ using Qt framework, OpenGL; VRPN, OpenMPI	[BBGS16]
large display + smartwatch	–	JavaScript, D3.js	[HBED18]
smart device (iPad)	–	ARKit2.0	[JXK+19]
smart device (iPhone, iPad)	–	React Native, ARKit	[CSW+20]
smart device (f.e. iPhone)	–	Unity, Vuforia, ARToolkit	[ESMT16]
Sony VAIO VGN-UX390N Ultra Mobile PC	GlobalSat BT-338 GPS, and InterSense InertiaCube3 (IC3) inertial orientation tracker	Goblin XNA, ARTag	[WF09]
(2) stereoscopic screens (for 3D), shutter glasses	3D cursor / 3D pick ray / virtual flashlight / 3D portals	Avango-Guacamole	[KWFK19]
Tablet (Sony Xperia)	–	Vuforia and Unity	[ETM+15]
zSapce 200 semi-immersive monitor, head tracking 3D glasses, 2D monitor	zSpace stylus pen (6DOF), computer mouse, keyboard	Java	[SJX+15]

Table A.3: Results for category: Aggregate

Input	Human Action	System Reaction	References
Controller	Intersecting elements	Remove intersection	[KFC+19], [KFC+19]
	Press buttons + bump hands	Merge	[ODZA18]
	Press button + move	Slice volume	[CBC+20]
	Touch | radial menu	Cluster data according to attributes	[IDJW19]
Multi-touch table	Touch | draw on the graph	Cluster	[BHM+18]
Self-made device	Move slider	Slice volume	[CBC+20]
Smartphone	Touch | draw shape	Group elements	[CSW+20]
+	Navigate Data | grab + throw	Remove from category	[ODZA18]
	Push UI button	Sort	[BHM+18]

Table A.5: Results for category: Change Modes

Input	Human Action	System Reaction	References
Audio	Voice command	Enable Navigate Data	[MBD+18], [FML+19], [MFM+19]
		Visualization change	[BTNB19]
		–	[LNQ+19], [FML+19]
Controller	Press button	Theta angle on / off	[WCST18]
	Swipe left to right	Change between modalities	[MGO+19]
Keyboard	Hold Control key	Enable scroll zoom	[WBR+20]
	Hold SHIFT key	Enable mouse navigation	[SJX+15]
		Enable z-axis scroll	[WBR+20]
	Press TAB key	Switch cursor (monitor - AR)	[WBR+20]
Leap Motion	Mid-air hand gesture	Visualization change	[CDH16]
Mouse	Double click	Change navigation mode	[SJX+15]
	Press scroll wheel	Switches z-axis translate - rotate	[WBR+20]
Sensors	Tilt device up / down	Show AR / scale map view	[WF09]

Table A.7: Results for category: Data Scroll

Input	Human Action	System Reaction	References
Controller	Press button	Move forward / backward in time	[BBGS16]
	Touch | circular motion	Navigate focus of timeline	[CWT18]
		Scroll through simulation steps	[KFC+19]
	Touch | horizontal swipe	Scroll through temporal evolution	[SWKA19]
	Touch | vertical swipe	Move between slice / targets / sizes	[MGO+19]
Keyboard	–	Scroll through code	[MSY+20]
Leap Motion	Gesture	Stepping through timeline / image stacks	[dJH+15]
Smart device	FoV | gesture	Altering menu items	[ESMT16]
Smartwatch	Touch | vertical swipe	Scrolling through content	[HBED18]
	Rotate crown	Scrolling through content	[HBED18]
Touch surface	Pan-touch	Scroll along route	[LSS17]

Table A.9: Results for category: Details-on-Demand

Input	Human Action	System Reaction	References
Audio	Voice command	Enable / disable visual cues	[MFM+19]
		Show / hide text	[LNQ+19]
		Shows parameters	[BTNB19]
Controller	Touch | tap	Details-on-demand	[FSN20]
Sensors	Ray tracing | gaze	Details-on-demand	[MGHK15], [CDH16]
	Ray tracing | head movement	Details-on-demand	[MBD+18], [MSY+20]
+	FoV | pick up object	Details-on-demand	[ESMT16]
	Navigate Data for details	Details-on-demand	[SFX19]
	Select for details	Details-on-demand	[SWKA19], [YLF+16], [HBED18], [LSS17]
	Touch element	Details-on-demand	[IDJW19], [WF09]

Table A.11: Results for category: Display Data

Input	Human Action	System Reaction	References
3D pick ray	Double click	Path tracing	[KWFK19]
Audio	Voice command	Show/hide / convert to miniature	[FML+19]
		Turn overlay on / off	[ERKL19]
Controller	Move sliders	Create bounding box	[CBC+20]
	Press button + move	Create bounding box	[CBC+20]
	Press button on object	Diffusion of odor	[PBE19]
	Touch | click	Automatic play through	[CWT18]
Keyboard	Press button	Freeze video	[WF09]
Leap Motion	Gesture	Play / pause videos	[dJH+15]
Multi-touch table	Press button	Colorize based on values	[BHM+18]
Sensors	Gaze | view map control	Show current location of interest	[WCST18]
	Gesture	Change color	[BTNB19]
		Custom simulation	[YLF+16]
		Raise / lower water levels	[YLF+16]
	Gesture | air tap	Visualize 3D structure	[FML+19]
Smart device	FoV | click on object	Show toggle buttons	[ESMT16]
	Wave device like brush	Distribute glyphs in 3D space	[CSW+20]
Smartwatch	Point to display + touch	Show preview on display	[HBED18]
+	Navigate Data + press button	Play/ pause playback	[SFX19]
	Navigate Data across array of axes	Axis swipe	[CCD+17]

Table A.13: Results for category: Edit Data

Input	Human Action	System Reaction	References
Controller	Press button	Place widget on the visualization	[MBD+18]
		Place marker / erase marker	[DCW+18]
Keyboard	Typing	Edit code	[MSY+20]
Multi-touch table	Push button	Add plot	[BHM+18]
Smart device	Data entry	Add data	[WWS20]
Smartwatch	Push gesture	Change, encode, reconfigure	[HBED18]
	Touch | crisscross motion	Delete selected set	[HBED18]
Tablet	Press button	Add / delete	[JXK+19]
Topographic map	Draw with finger	Sketch route	[LSS17]
+	Grab + place | throw metaphor	Discard axis	[CCD+17]
	Throw visualization to the ground	Delete	[LCPD19]
	Wiggling axis part of visualization	Move elements with axis	[CCD+17]

Table A.15: Results for category: Edit Selection

Input	Human Action	System Reaction	References
3D pick ray	Ray tracing | intersecting	Selection	[KWFK19]
Audio	Voice command	Confirm selection	[BTNB19]
		Select menu items / labels	[ERKL19]
Controller	Clicker button	Select menu items / labels	[ERKL19]
	Press button	Clear selection	[BBGS16]
		Confirm selection	[JKJ+19]
		Selection	[MGO+19]
	Ray tracing | intersecting	Add to selection	[HRD+19]
		Brush selection	[BBGS16], [HRD+19]
		Choose selection	[JKJ+19]
		Remove from selection	[HRD+19]
		Selection	[SWKA19]
	Tap element	Selection	[DCW+18]
	Touch | double tap	Selection	[FSN20]
Mouse	Click on element	Selection	[SJX+15]
Sensors	Ray tracing | gaze	Choose selection	[MBD+18], [BTNB19]
		Selection	[LNQ+19], [FML+19], [CCG19]
	Gesture | air tap	Confirm selection	[MBD+18]
		Selection	[YLF+16], [MFM+19], [BMD18], [BMD18]
Smart device	FoV | move to object	Select and scan	[ESMT16]
Smartwatch	Touch | tapping / circling	Selection	[HBED18]
	Touch-and-hold	Temporary selection	[HBED18]
	Gesture + tracing | pointing	Choose selection	[HBED18]
Stylus pen	Gesture + tracing | pointing	Selection	[SJX+15]
Tablet	Gesture + tracing | pointing	Choose selection	[ETM+15]
	Press buttons	Confirm selection / deselection	[ETM+15]
+	Stop Data Scroll on element	Select checkpoint	[LSS17]
	Option Control | brush	Brush	[LCPD19]

Table A.17: Results for category: Filter

Input	Human Action	System Reaction	References
Audio	Voice command	Show / hide visualization	[MBD+18]
Controller	Press button	(De)select variable range	[BBGS16]
	–	Time / visual filtering	[CDH16]
Leap Motion	Virtual keyboard	Filtering / dynamic queries	[MGHK15]
Sensors	Gesture | air tap	Information becomes visible	[CCG19]
		Photo displayed as query on canvas	[BMD18]
Smartwatch	Selected item | rotate crown	Filter	[HBED18]
Tablet	Press button	Create queries	[ETM+15]
+	Aggregate (cluster) to filter	Filter	[BHM+18]
	Apply widget option to filter	Filter	[CCD+17]
	Press virtual widget button	Filter based on attributes	[MFM+19]
	Put boxes side by side	Combine, compare, sort information	[ESMT16]
	Select to filter	Display query results	[BMD18]
		Filter	[FSN20], [MFM+19]

Table A.19: Results for category: Navigate Data

Input	Human Action	System Reaction	References
3D pick ray	Pick ray on element	Drag element	[KWFK19]
Controller	Joystick	Zoom	[NWZ+16]
	Gesture	Rotate	[FSN20]
		Scale	[FSN20]
		Translate	[FSN20]
	Press button + move	Grab	[MGO+19], [CCD+17], [SFX19], [ODZA18]
		Move	[CCD+17], [ODZA18]
		Rotate	[BBGS16], [MGO+19], [HRD+19]
		Scale	[BBGS16], [HRD+19]
		Translate	[BBGS16], [HRD+19]
		Zoom	[ODZA18]
	Touch	Pan	[WWS20], [CWT18]
		Rotate	[SWKA19]
		Zoom	[CWT18], [SWK+16]
	–	Move	[SWK+16], [LCPD19]
Leap Motion	Gesture	Zoom	[NWZ+16], [LSS17]
Mouse	Press left button	Drag view	[SJX+15]
		X-y-axis rotate	[WBR+20]
	Press right button	X-y-axis translate	[WBR+20]
	Scroll wheel	Z-axis move cursor	[WBR+20]
		Z-axis rotate	[WBR+20]
		Zoom	[WBR+20]
	–	Zoom / pan	[NWZ+16]
Multi-touch table	Press button	Rotate / move	[BHM+18]
Sensors	Air tap + gesture	Rotate	[MSY+20]
	Air tap + move body	Translate	[MSY+20]
	Gesture	Grab	[KFC+19], [JKJ+19]
		Move	[YLF+16], [FML+19], [MFM+19], [JKJ+19], [BTNB19]
		Rotate	[MBD+18], [dJH+15], [YLF+16], [FML+19], [NYW+16], [MFM+19], [JKJ+19], [BTNB19]
		Scale	[MBD+18], [YLF+16], [FML+19], [KFC+19], [MFM+19], [JKJ+19], [IDJW19], [BTNB19]
	Gesture | air tap	Move	[MBD+18]
		Place 2D canvas	[BMD18]
		Rotate	[LNQ+19]
		Zoom	[LNQ+19]
	Touch | drag-and-drop	Move	[WF09]
Smart device	FoV touch gesture	Move (2D plane)	[CSW+20]
		Scale	[CSW+20]
		Zoom	[JXK+19], [ESMT16]
	Raise / lower device	Move up / down	[CSW+20]
Stylus pen	Press button + move	Rotate / zoom	[SJX+15]

Table A.21: Results for category: Navigate View

Input	Human Action	System Reaction	References
Controller	Hold at certain distance	Determine speed of movement	[DCW+18]
	Press button on object	Move miniature camera	[DCW+18]
	Ray tracing + press button	Move viewpoint towards position	[DCW+18]
		Put viewpoint on object position	[SWKA19]
		Put viewpoint on position	[DCW+18], [HRD+19]
	Touch input	Fly through visualization	[SWKA19]
		Project range	[DCW+18]
	Use standard teleport	Put viewpoint on position	[SFX19], [SWK+16]
Keyboard	WASD / arrow keys	–	[SJX+15]
Mouse	Press mouse buttons	Move forward / backwards	[SJX+15]
	–	Manually modify view	[MGO+19]
Sensors	Gesture | vertical	Change viewpoint	[CDH16]
	Ray tracing | gaze	Determine direction	[CDH16]
Stylus pen	Point and press button	Move viewpoint towards position	[SJX+15]
	Press button	Rotate viewpoint	[SJX+15]

Table A.23: Results for category: Open

Input	Human Action	System Reaction	References
Audio	Voice command	Change dataset	[LNQ+19]
Controller	Hold and release visualization	Place visualization	[LCPD19]
Smartwatch	Touch | swipe away from elbow	Data transfer (smartwatch to display)	[HBED18]
	Touch | swipe towards elbow	Data transfer (display to smartwatch)	[HBED18]

Table A.25: Results for category: Option Control

Input	Human Action	System Reaction	References
Audio	Voice command	Copy dataset / reset view	[BTNB19]
		Share visualization	[MFM+19]
		Use UI buttons / shift window	[ERKL19]
Controller	Clicker button	Press UI buttons / shift window	[ERKL19]
	Drag copy widget on view	Duplicate view	[KFC+19]
	Move controller to widget	Widget options	[CCD+17]
	Press button	Change attributes	[MFM+19]
		Change location of interest	[WCST18]
		Create ray for selection	[HRD+19]
		Go back to previous view	[ODZA18]
		Request more simulation results	[BBGS16]
		Reset view	[BBGS16]
	Press button + move controller	Adjust value	[CBC+20]
		Modify culling volume	[KFC+19]
	Slide physical knob on slider	Adjust value	[CBC+20]
	Touch — radial menu	Change variables	[KFC+19]
		Use option	[LCPD19]
Keyboard	Typing	Execute code, access examples panel	[MSY+20]
Multi-touch table	Push button	Use option	[BHM+18]
Sensors	Gesture — air tap	Reset view	[CCG19]
	Gesture + tracing | pointing	Change visualization parameters	[NYW+16]
	Natural gesture	Feedback	[JXK+19]
	Ray tracing — gaze	Determine focused element	[WCST18]
	Ray tracing — head movement	Move cursor	[ERKL19]
Smart device	Touch | UI elements	Create visual mapping	[CSW+20]
		Specify ratio between options	[ETM+15]
		Visualize menu	[ESMT16]
Smartwatch	Touch | double-tap	Enable ”distant interaction”	[HBED18]
	Touch | tap element	Option: configure dimensions, scales	[HBED18]
+	Navigate Data | use UI	Change parameters	[LCPD19]
	Navigate Data | use UI slider	Change network threshold	[JKJ+19]
	Select menu item	Use option	[JKJ+19]

Table A.27: Results for category: Save

Input	Human Action	System Reaction	References
Audio	Voice command	Records current configuration	[BBGS16]
Controller	Grab + pull gesture on visualization	Create copy	[LCPD19]
Keyboard	Press key	Save image	[WF09]

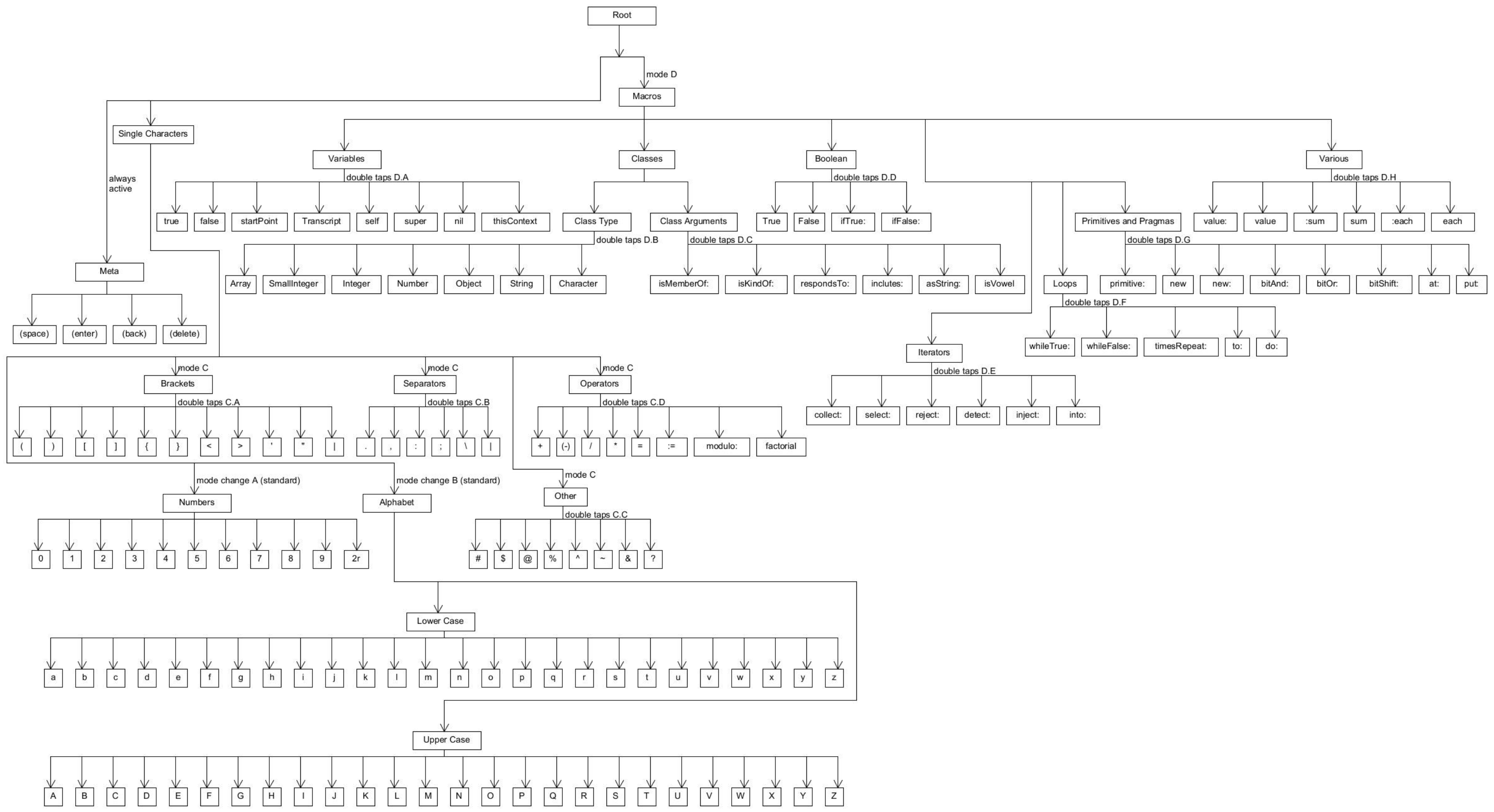


Figure A.3: Initial Idea for implementation of Mapping B

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

place, date, signature