Toward Agile Situated Visualization: An Exploratory User Study

Leonel Merino
University of Stuttgart
leonel.merino@visus.uni-stuttgart.de

Alexandre Bergel
ISCLab, Department of Computer Science, University of Chile
abergel@dcc.uchile.cl

Boris Sotomayor-Gómez
Ernst Strüngmann Institute for Neuroscience in Cooperation with Max Planck Society
boris.sotomayor@esi-frankfurt.de

Michael Sedlmair
University of Stuttgart
michael.sedlmair@visus.uni-stuttgart.de

Xingyao Yu
University of Stuttgart
xingyao.yu@visus.uni-stuttgart.de

Daniel Weiskopf
University of Stuttgart
daniel.weiskopf@visus.uni-stuttgart.de

Ronie Salgado
University of Chile
roniesalg@gmail.com

Abstract

We introduce *AVAR*, a prototypical implementation of an agile situated visualization (SV) toolkit targeting liveness, integration, and expressiveness. We report on results of an exploratory study with AVAR and seven expert users. In it, participants wore a Microsoft HoloLens device and used a Bluetooth keyboard to program a visualization script for a given dataset. To support our analysis, we *(i)* video recorded sessions, *(ii)* tracked users' interactions, and *(iii)* collected data of participants' impressions. Our prototype confirms that agile SV is feasible. That is, *liveness* boosted participants' engagement when programming an SV, and so, the sessions were highly interactive and participants were willing to spend much time using our toolkit (*i.e.*, median $\geq$ 1.5 hours). Participants used our *integrated* toolkit to deal with data transformations, visual mappings, and view transformations without leaving the immersive environment. Finally, participants benefited from our *expressive* toolkit and employed multiple of the available features when programming an SV.

Author Keywords

Situated Visualization; Augmented Reality; User Study;

CCS Concepts

•**Human-centered computing** $\rightarrow$ **Visualization design and evaluation methods;** *Mobile devices;*

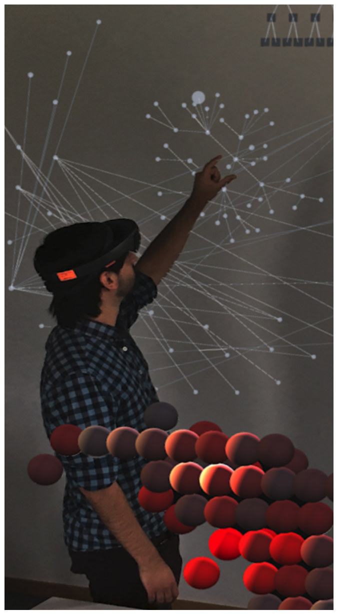


Figure 1: A user wears a Microsoft HoloLens device to interact with an SV.

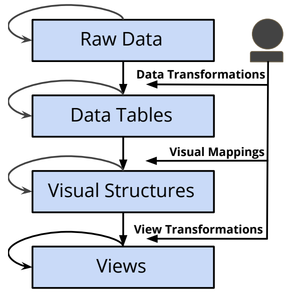


Figure 2: The 3-process reference model of visualization.

Introduction

Situated visualization (SV) promotes interactive analytical reasoning by embedding data visualizations in the physical environment through immersive augmented reality (AR) [8] (see Figure 1). Users can interact with an SV using the third spatial dimension, which stimulates cognitive aspects such as engagement, embodiment, and recall [5, 10]. To boost reasoning, an SV toolkit has to support users in quickly building visualizations that combine real objects with visual representations of data. Yet, we observe that existing SV toolkits lack such agility, which hinders the applicability of SV in practice.

To enable users to create and modify data visualizations situated in a real context in an agile fashion, we hypothesize that an SV toolkit must offer *(i) live* feedback when they create or modify a visualization, *(ii)* an *integrated* environment that supports all processes involved in visualization, and *(iii) expressive* means to support increment visualization design. We implemented a prototype that supports agile SV, which we call *AVAR*.

We conducted an exploratory study with seven expert users and carefully analyzed their behavior when creating an SV. We observed that *live* feedback boosted participants' engagement when programming an SV, and so, they were highly interactive and willing to spend long time spans (*i.e.*, median $\geq$ 1.5 hours) performing incremental modifications to visualization scripts. Our *integrated* toolkit allowed participants to deal with the visualization as a whole without leaving the immersive environment. Finally, our *expressive* toolkit allowed participants to employ multiple available features when programming an SV.

Our contributions are *(i)* an exploratory user study and *(ii)* an open source prototype released under MIT license, thus making it fully available to practitioners and researchers[1].

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

Related work

There is a lack of toolkits to guide authoring SVs that offer ready-to-use building blocks to speed up development [10]. Amongst the few existing ones, none of them focuses on agility (*i.e.*, incremental visualization construction). For instance, SiteLens [13] is a situated analytics system for supporting site visits in urban planning. In it, users can visualize an already curated dataset with a limited number of techniques. Munin [1] is a middleware for ubiquitous analytics that focuses on large scale distributed visualization for collaborative environments. In it, visualizations can be displayed in mediums such as wall displays, smartphones, and tabletops, but not in immersive devices for AR (*e.g.*, Microsoft HoloLens). VRIA [4] is a Web-based framework for creating immersive analytics experiences that involves a programming language with limited capabilities of expressiveness and reflection, posing a barrier for data transformations.

As opposed to our SV toolkit, existing toolkits for immersive analytics such as NiwViw [14], DXR [12], IATK [6], and ImAxes [7] support a limited number of fixed and ready-to-use templates for visualization techniques, impairing expressiveness. These toolkits offer only partial integration, and so, they require users to perform data transformations using a desktop computer, which hinders agility.

Our approach differs from previous works as it targets users with programming knowledge. This fundamental difference can explain limitations of existing authoring toolkits. To the best of our knowledge, our SV approach is the first one that supports the three processes of programming an interactive visualization (shown in Figure 2). We consider that integrating support for all these processes in a live and expressive programming environment can lead to agile SV.

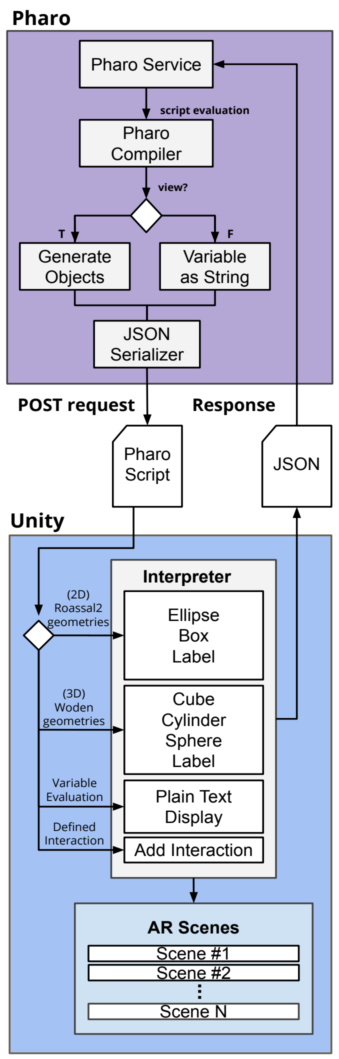


Figure 3: The distributed architecture of AVAR.

AVAR: An Agile Situated Visualization Toolkit

Agile SV is a highly dynamic iterative process for exploring multiple facets of a dataset. In it, for instance, users can add previously filtered data to a view and analyze how it changes without having to leave the immersive environment (*e.g.*, removing an AR headset): script building, data exploration, and visualization exploitation happen in the AR environment. We identify three main challenges for agile SV: *(C.1)* an infrastructure that supports live programming, *i.e.*, short feedback loop when evaluating a (visualization) script, *(C.2)* an integrated immersive environment in which users can type and read visualizations scripts, and *(C.3)* a language that is sufficiently expressive to define multiple visualization designs but simple enough to be used in immersive AR.

We introduce *AVAR*, our prototypical implementation for agile SV. Figure 3 presents a diagram with the software stack and the components used in the implementation of AVAR. We employ a Microsoft HoloLens and a Bluetooth keyboard as input/output devices that support user interaction, see Figure 4. Figure 5 shows a virtual panel that enables users to type visualization scripts, load visualization examples, and receive error notifications.

In the design of AVAR, we maximize reusing existing tools. As a consequence, the implementation phase mainly consists of integrating these third-party tools in the immersive AR environment. We integrate a fully operational programming language into an immersive environment. To this end, we adopt Pharo[2], a modern implementation of Smalltalk. Pharo is a dynamically typed message passing language that has an expressive syntax that allows users to perform complex operations by typing short scripts [2]. Pharo is interpreted, that is, users do not have to wait for a script to compile, but they can evaluate scripts in a live environment.

[2]http://pharo.org/, accessed 15.12.2019

Pharo is highly reflective, which eases integration to external environments. Users can define data visualizations using Roassal2[3] and Woden[4], 2D and 3D data visualization engines, respectively. These engines implement multiple 2D and 3D visualization techniques that are shipped out-of-the-box such as parallel coordinates, treemaps, node-link diagrams, and space-time cube matrices. Finally, we implement a thin application in Unity 3D[5] that communicates with Pharo as a backend, handles user interaction, and renders a graphical user interface in AR. Our toolkit supports the following 3-step process:

1. *Data Transformations*. To build a data visualization, users first apply several transformations to a given raw dataset to create data tables, *e.g.*, filtering, formatting, normalizing. As these transformations are available in Pharo, users have access to multiple functionalities for data transformations. To the best of our our knowledge, this process is not fully supported by existing SV toolkits.
2. *Visual Mappings*. Next, users choose visual mappings to apply to data tables toward creating visual structures. To this end, we rely on multiple existing "builders" in Roassal2 and Woden, which are domain-specific languages (DSLs) that support the rapid construction of particular interactive visualizations.
3. *View Transformations*. Finally, a view is rendered in immersive AR to which users apply various view transformations (programmatically as well as using natural user interfaces). For instance, through hand gestures users can rotate a view to obtain a different perspective. Users can also combine a hand gesture with walking to relocate a view to a different place.

[3]https://github.com/ObjectProfile/Roassal2, accessed 15.12.2019
[4]https://github.com/ronsaldo/woden, accessed 15.12.2019
[5]https://unity3d.com/, accessed 15.12.2019

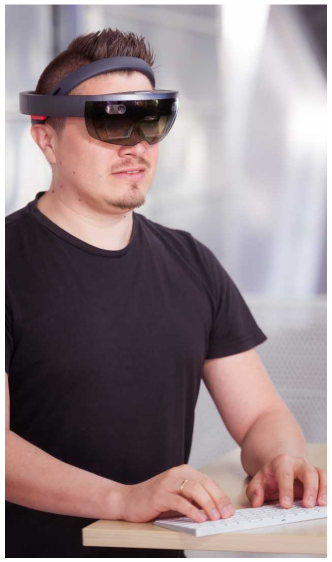
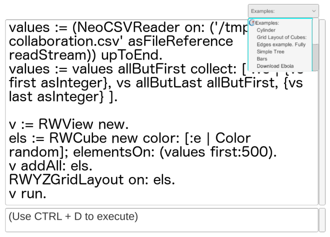


Figure 4: In the study, participants wore a Microsoft HoloLens device and used an Apple Magic Bluetooth keyboard.



```
values := (NeoCSVReader on: ('/tmp
collaboration.csv' asFileReference
readStream)) upToEnd.
values := values allButFirst collect: [
first asInteger}, vs allButLast allButFirst, {vs
last asInteger} ].

v := RWView new.
els := RWCube new color: [:e | Color
random]; elementsOn: (values first:500).
v addAll: els.
RWYZGridLayout on: els.
v run.
```

(Use CTRL + D to execute)

Figure 5: The graphical user interface in AR of AVAR: (center) code editor, (upper-right corner) examples browser, and (bottom) console panel.

Exploratory User Study

We adopted a template previously introduced [11] to describe the scope of our study:

> We examine the usage of the AVAR toolkit for authoring *situated visualizations* displayed in a *Microsoft HoloLens* device in the context of *an exploratory analysis* from the point of view of *expert users*.

Pilot We used a pilot study with two participants to fine-tune the study factors: task and datasets that could be understood quickly, and the inclusion of more examples to demonstrate the capabilities of visualization engines.

Participants As our prototype uses a Smalltalk scripting language, we decided to conduct our study at ESUG'19[6]. We sent an open invitation through the conference mailing list. In the end, we scheduled sessions with seven participants, who were not paid and freely opted to participate in the study. All participants were male. Their median age was 31 $\pm$ 8.5 years, and they had a considerable experience using Smalltalk (*i.e.*, experience $\geq$ 6 years). We also asked for their familiarity with the technologies involved in the study. In summary, participants declared to *(i)* frequently build data visualizations, *(ii)* know little of the API of the 2D visualization engine, *(iii)* do not know details of the API of the 3D visualization engine, and *(iv)* have used a device like the Microsoft HoloLens no more than once.

Dataset We selected two datasets used in previous studies [12, 7]. One dataset contains co-authorship information along a period of time, which we considered adequate to minimize the complexity of Task 1 (*e.g.*, it has 209 items and 4 properties). In Task 2, participants used a second dataset that contains 6,497 samples of wine (1,600 red and 4,897 white) described by 12 data attributes. Both datasets are publicly available[7,8].

Tasks *Task 1.* We asked participants to build a space-time cube visualization (results shown in Figures 6). Each cube represents the relation between two co-authors. The X and Z axes (co-planar to the room's floor) represent the list of authors, and the Y axis represents time. Time is over-loaded in the color of the cubes, which use a color ramp from blue to yellow. To clarify the given task, we handed to participants a printout of the expected resulting visualization. *Task 2.* As a second, and optional task, we asked participants to analyze main differences between white and red wine based on the given dataset. To this end, participants were encouraged to use the multiple features available in the visualization engines.

Apparatus Participants wore a Microsoft HoloLens 1 head-set. The headset was complemented with an Apple Magic Bluetooth keyboard (see Figure 4). Participants used the keyboard to interact with the three main sections of the graphical user interface: *(i)* code editor, *(ii)* examples browser, and *(iii)* console panel (as shown in Figure 5). Participants could scroll through the code either using the keyboard or a hand gesture. Participants could interact with visualizations in three ways by *(1)* hovering over an element using head movements to obtain contextual information, *(2)* rotating a visualization using an airtap and hold combined with an horizontal hand gesture, and *(3)* translating the visualization to a new location by using an airtap hand gesture and body and head movements.

Procedure The sessions with each participant were conducted in a quiet room. The room had enough space for

[6]European Smalltalk Users Group, accessed December 15, 2019, https://esug.github.io/2019-Conference/conf2019.html

[7]https://github.com/ronellsicat/DxR/blob/master/Assets/StreamingAssets/DxRData/collaboration.csv, accessed 15.12.2019

[8]http://www3.dsi.uminho.pt/pcortez/wine/winequality.zip, accessed 06.01.2020

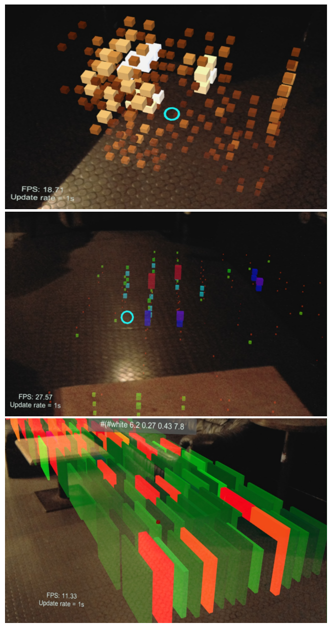


Figure 6: Visualizations by participants in the user study.

Simulator Sickness	Rating
General discomfort	Moderate
Fatigue	Slight
Headache	None
Eye strain	Slight
Difficulty focusing	Slight
Increased salivation	None
Sweating	None
Nausea	None
Difficulty concentrating	None
Fullness of the head	Slight
Blurred vision	Slight
Dizzy (eyes open)	None
Dizzy (eyes closed)	None
Vertigo (Giddiness)	None
Stomach awareness	None
Burping	None

Table 1: Median ratings using the Simulator Sickness questionnaire.

the participants to move around. The room had a high table for stand-up coding and a normal table for participants who preferred to sit on a chair. At the beginning of the session, participants were asked to read and sign a consent form that informed them of the characteristics of the study. Next, the experimenter read an introduction to explain the details of the phases in the study. We encouraged participants to share their thoughts using a think-aloud protocol. Participants were free to stop the session at any time. Once participants finished a task, we asked them about the perceived difficulty of the task. To examine the (lack of) comfort experienced by participants, we asked them to fill in a Simulator Sickness questionnaire [9]. To assess the perceived usability of our system, participants were asked to fill in a System Usability Scale questionnaire [3].

Data Collection We *(i)* video recorded the sessions (*i.e.*, 13 hours and 30 minutes), *(ii)* tracked events of participants' interactions with the graphical user interface and with the environment (*i.e.*, 1029 interaction events), and *(iii)* collected filled-in questionnaires (*i.e.*, 28 pages in total).

Results

A set of charts that summarize the results of the study is presented in Figure 8. Due to the limited space, we opted to present only the results of *Task 1*, even though 5 participants also solved *Task 2*. Charts are sorted by time (*e.g.*, participant 1, who had the longest session, is presented at the top). A horizontal black bar, at the middle of each chart, encodes the length of a session. Such bars split charts into two sections. In the upper section, gray circles are vertically arranged to indicate which area in the graphical interface has the focus of a participant. A green circle indicates a script that is successfully executed, otherwise, the circle is red. An additional horizontal bar encodes, using three colors, which visualization process participants are addressing. The lower section of a chart supports a temporal analysis of visualization scripts: Vertical bars (in green) show additions and (in red) deletions of code. Additional circles depict the total size of a script at certain points in time. Circles are connected with lines to indicate the evolution of the script size. The median values of the ratings of participants using the Simulator Sickness questionnaire are presented in Table 1 and results of the System Usability Scale (SUS) questionnaire are presented in Figure 7. The median SUS score by participants was 58, with a maximum of 70 and a minimum of 53. The median rate at which participants interacted with our system was 1.5 $\pm$ 0.7 events per minute.

Discussion

AVAR confirms the feasibility of an agile SV toolkit based on liveness, integration, and expressiveness. All participants were able to solve the first task, even though they experienced moderate discomfort wearing the headset for a long period of time. Participants agreed with the observation that "most people would learn to use the system quickly" as they did not need to learn lots of things to get going with the system. However, they considered that AVAR requires improvements to be easy-to-use.

Liveness. More experienced users (*e.g.*, 6 and 7) solved the Task 1 with fewer interactions in a shorter time than less experienced users (*e.g.*, 1 and 2). We also observe that our live environment boosted participants engagement, as five out of seven participants were willing to solve the second (optional) task. All sessions were highly interactive. In agile SV, users rely on liveness to obtain feedback of script executions, which indeed, were uniformly distributed in time (see Figure 8).

Integration. Participants engaged in long experimental sessions that lasted a median of 106 minutes$\pm$27.5 (with a maximum of 148 minutes and a minimum of 82 minutes). In them, participants were able to deal with all the processes

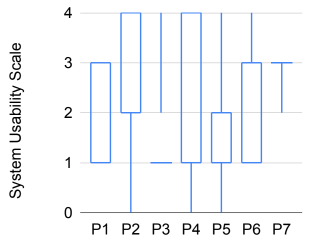


Figure 7: Ratings by participants using the System Usability Scale questionnaire.

Acknowledgments
Merino, Sedlmair, Sotomayor-Gómez, and Weiskopf acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 251654672 – TRR 161. Bergel thanks LAM Research for its financial support. Sotomayor-Gómez thanks Universidad Austral de Chile and Universidad de Chile for their financial support.

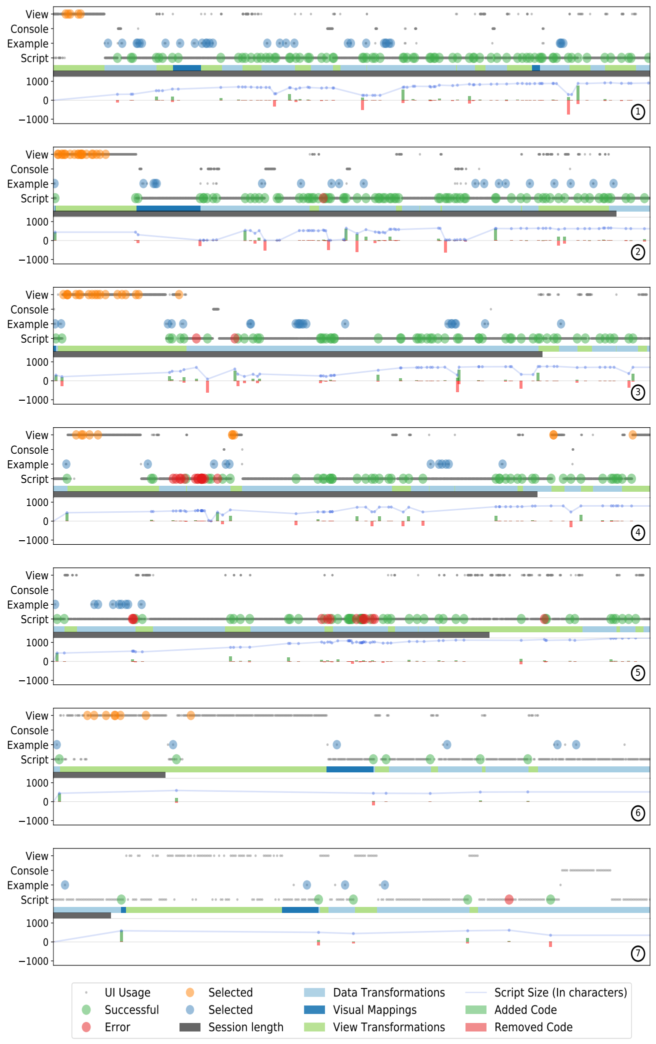


Figure 8: Charts that present the interactions of the seven participants who used our SV toolkit.

required to solve the tasks without leaving the immersive environment. In agile SV, users require to address visualization processes in a continuous loop (and not sequentially) as show in Figure 8.

Expressiveness. Our included expressive language enabled participants to use various features that they found amongst the visualization engines available. For instance, participants 1 and 3 used `RTTabTable` to manipulate data tables, participants 2 and 7 used `RWElement` for handling 3D elements, participant 2 used `RWAlign` to layout elements in a 3D space, participant 3 used `RWCylinder` to produce cylinders as visual elements, and participant 7 used `RTScale` to scale elements and maximize the use of the available space in the room. Other features used by all participants were `RWView` to specify views, `RWXZGridLayout` to layout elements as a 3D grid, and `RWCube` to define cube shapes for elements. In agile SV, users depend on having multiple features available to express SV designs in an iterative fashion.

Conclusion
We introduce *AVAR*, an agile SV toolkit based on liveness, integration, and expressiveness. We report on an exploratory study with seven expert users who were asked to program an SV script. We analyzed how our design choices impacted participants' behavior. We found that live feedback boosted the engagement of the participants, who worked highly interactively and were willing to spend much time using our toolkit. Participants were able to deal with visualization as a whole without leaving the immersive environment. Finally, we observed that participants employed multiple of the available features when programming an SV. In the future, we plan to improve our design, investigate other means for interaction, and conduct further evaluations.

REFERENCES

[1] Sriram Karthik Badam, Eli Fisher, and Niklas Elmqvist. 2014. Munin: A peer-to-peer middleware for ubiquitous analytics and visualization spaces. *IEEE Transactions on Visualization and Computer Graphics* 21, 2 (2014), 215–228.

[2] Alexandre Bergel. 2016. *Agile Visualization*. LULU Press. `http://AgileVisualization.com`

[3] John Brooke. 1996. Usability Evaluation in Industry. In *SUS: A quick and dirty usability scale*, B. Thomas Patrick W. Jordan, Ian Lyall McClelland, and Bernard Weerdmeester (Eds.). Vol. 189. CRC Press, Chapter 21.

[4] Peter William Scott Butcher, Nigel W. John, and Panagiotis D. Ritsos. 2020. VRIA: A Web-based framework for creating immersive analytics experiences. *IEEE Transactions on Visualization and Computer Graphics* (2020).

[5] Stuart K. Card and Jock Mackinlay. 1997. The structure of the information visualization design space. In *Proc. of INFOVIS*. IEEE, 92–99.

[6] Maxime Cordeil, Andrew Cunningham, Benjamin Bach, Christophe Hurter, Bruce H. Thomas, Kim Marriott, and Tim Dwyer. 2019. IATK: Immersive analytics toolkit. In *Proc. of IEEE VR*. IEEE, 21–31.

[7] Maxime Cordeil, Andrew Cunningham, Tim Dwyer, Bruce H. Thomas, and Kim Marriott. 2017. ImAxes: Immersive axes as embodied affordances for interactive multivariate data visualisation. In *Proc. of UIST*. ACM, 71–83.

[8] Neven A.M. ElSayed, Bruce H. Thomas, Kim Marriott, Julia Piantadosi, and Ross T. Smith. 2016. Situated analytics: Demonstrating immersive analytical tools with augmented reality. *Journal of Visual Languages & Computing* 36 (2016), 13–23.

[9] Robert S. Kennedy, Julie M. Drexler, Daniel E. Compton, Kay M. Stanney, D. Susan Lanham, and Deborah L. Harm. 2003. Configural scoring of simulator sickness, cybersickness and space adaptation syndrome: similarities and differences. In *Virtual and Adaptive Environments: Applications, Implications, and Human Performance Issues*, Lawrence J. Hettinger and Michael W. Haas (Eds.). CRC Press, Chapter 12.

[10] Kim Marriott, Falk Schreiber, Tim Dwyer, Karsten Klein, Nathalie Henry Riche, Takayuki Itoh, Wolfgang Stuerzlinger, and Bruce H. Thomas. 2018. *Immersive Analytics*. Vol. 11190. Springer.

[11] Leonel Merino, Mohammad Ghafari, Craig Anslow, and Oscar Nierstrasz. 2018. A systematic literature review of software visualization evaluation. *Journal of Systems and Software* 144 (2018), 165–180.

[12] Ronell Sicat, Jiabao Li, JunYoung Choi, Maxime Cordeil, Won-Ki Jeong, Benjamin Bach, and Hanspeter Pfister. 2019. DXR: A toolkit for building immersive data visualizations. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 715–725.

[13] Sean White and Steven Feiner. 2009. SiteLens: situated visualization techniques for urban site visits. In *Proc. of CHI*. ACM, 1117–1120.

[14] Dianna Yim, Alec McAllister, Caelum Sloan, Rachel Lee, Steven Vi, Teresa Van, Wesley Willett, and Frank Maurer. 2018. NiwViw: Immersive analytics authoring tool. In *Proc. of ISS*. ACM, 425–428.