

Flex-ER: A Platform to Evaluate Interaction Techniques for Immersive Visualizations

María-Jesús Lobo, Christophe Hurter, Pourang Irani

▶ **To cite this version:**

María-Jesús Lobo, Christophe Hurter, Pourang Irani. Flex-ER: A Platform to Evaluate Interaction Techniques for Immersive Visualizations. Proceedings of the ACM on Human-Computer Interaction , 2020, 4 (ISS), pp.195. 10.1145/3427323 . hal-03094867v2

HAL Id: hal-03094867

https://enac.hal.science/hal-03094867v2

Submitted on 15 Jun 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Flex-ER: A Platform to Evaluate Interaction Techniques for Immersive Visualizations

MARÍA-JESÚS LOBO, LASTIG, Univ Gustave Eiffel, ENSG, IGN, France
CHRISTOPHE HURTER, University of Toulouse, ENAC, France
POURANG IRANI, University of Manitoba, Canada

Extended Reality (XR) systems (which encapsulate AR, VR and MR) is an emerging field which enables the development of novel visualization and interaction techniques. To develop and to assess such techniques, researchers and designers have to face choices in terms of which development tools to adopt, and with very little information about how such tools support some of the very basic tasks for information visualization, such as selecting data items, linking and navigating. As a solution, we propose Flex-ER, a flexible web-based environment that enables users to prototype, debug and share experimental conditions and results. Flex-ER enables users to quickly switch between hardware platforms and input modalities by using a JSON specification that supports both defining interaction techniques and tasks at a low cost. We demonstrate the flexibility of the environment through three task design examples: brushing, linking and navigating. A qualitative user study suggest that Flex-ER can be helpful to prototype and explore different interaction techniques for immersive analytics.

CCS Concepts: • **Human-centered computing** → **User studies**; **Empirical studies in HCI**; *Empirical studies in visualization*; Mixed / augmented reality; Virtual reality.

Additional Key Words and Phrases: immersive analytics; extended reality; user studies

ACM Reference Format:
María-Jesús Lobo, Christophe Hurter, and Pourang Irani. 2021. Flex-ER: A Platform to Evaluate Interaction Techniques for Immersive Visualizations. 1, 1 (June 2021), 20 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Immersive analytics is an emerging field, that provides new opportunities and challenges for data visualizations. Recent applications include trajectory visualization [30], network visualization [15], and geographic visualization [65]. The development of this field is supported by recent advances in eXtended Reality systems (XR), that encapsulate Augmented Reality, Virtual Reality, and Mixed Reality. Affordable hardware platforms, such as VR head-mounted displays, head mounted augmented reality devices or mobile phones, now use powerful processors that can render high resolution images at high frame rates. However, even if some empirical studies have compared interaction techniques for immersive analytics [5, 53], it remains a challenge to evaluate the capabilities of these platforms [42]. No clear design guidelines exist yet, making it difficult for a designer to choose which tool to use for various tasks.

Authors' addresses: María-Jesús Lobo, LASTIG, Univ Gustave Eiffel, ENSG, IGN, F-94160 Saint-Mande, France, maria-jesus.lobo@ign.fr; Christophe Hurter, christophe.hurter@enac.fr, University of Toulouse, ENAC, 7, Av. E. Belin, Toulouse, France; Pourang Irani, University of Manitoba, Winnipeg, Canada.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
Manuscript submitted to ACM

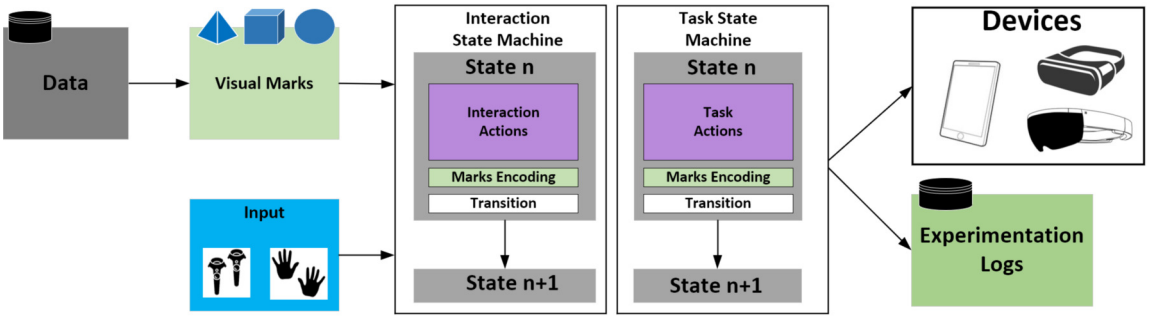


Fig. 1. Schematic pipeline of Flex-ER depicting the components considered.

Most immersive analytics applications include some sort of interaction technique, to support tasks such as navigation, selection, or filtering. However, these applications do not always evaluate these interaction techniques, and the existing evaluation results are often tied to specific hardware and software platforms. In a recent survey on immersive analytics (IA), the second major challenge presented by Fonnet and Prie [26], "Coverage towards best practices", focuses on the need for sharing IA user experiences and the lack of evaluations targeting low-level tasks. Furthermore, given the empirical evidence of perceptual differences between devices [16], we argue that there is a need for replicating user study results to advance the burgeoning field of immersive analytics.

We introduce Flex-ER, an environment designed to ease sharing immersive analytics interaction techniques and user studies. Flex-ER enables users to prototype, test and conduct user studies for immersive data analytics. It groups the definition of the data visualization, the interaction techniques and the experimental tasks in different components using a single JSON file, inspired by the Vega specification [58]. It is based on a modular approach and relies on the WebVR technology, to enable users to choose from a wide array of platforms to test the interaction techniques and the visualizations, while keeping the same tasks, all in the same interface. We expect Flex-ER to alleviate the burden of having to design new experimental platforms, and to create a common ground to evaluate immersive interactive visualizations across devices.

In summary, the contributions of this paper, framed according to toolkit goals [37] are:

- a JSON specification to define the data, interaction techniques and tasks for IA user studies, designed to reduce authoring time and complexity;
- Flex-ER, a user interface designed to ease the prototyping and debugging of user studies defined by the JSON specification, that aims at enabling replication and creative exploration of experimental conditions;
- an application programming interface (API) to extend the JSON specification;
- a preliminary user evaluation to assess the utility of Flex-ER environment based on a demonstration and a system walk-through.

2 RELATED WORK

2.1 Authoring interactive visualizations

There has been a lot of recent efforts to facilitate authoring visualizations. For example, the data-driven guides [31] and Lyra [56] enable users to create 2D visualizations without needing to write code. VegaLite [57] introduces a declarative grammar that describes visualizations. Reactive Vega [58] extends this model by adding interaction and

view composition. Other examples that support data interactions are Tableau, ggplot and d3 [7]. VizDock [13] enables users to add interaction to web-based visualization using a toolbar.

Some of the approaches mentioned above have inspired toolkits designed specifically for visualizing data in extended reality. IATK [14], DXR [59] and VRIA [10] are based on grammars of graphics to define immersive visualizations, using either Unity or web-based technologies. PapARVis Designer [12] enables creating augmented static visualizations.

Glance [25] is a toolkit to create graphics based visualizations. The focus of these toolkits is to facilitate the design of immersive visualizations. Flex-ER also supports defining a set of basic visualizations but its main goal is to support defining and sharing user studies comparing immersive interaction techniques and visualizations, rather than supporting a wide range of visualization types. More generally, some general purpose lower level APIs such as Three.js or its lower entity abstraction layer A-frame, Babylon.js or advanced game engines such as Unity enable creating immersive visualizations.

High level specifications have also been explored to describe interaction techniques, based for example on data-flows [48]. Unit [48] abstracts the interaction layer from an application by specifying the dataflow. Broll et al. [9] present a component based infrastructure to prototype interactive AR applications. Aside from data flows, the state machine approach plays an important role when defining interactions [3, 6, 49]. Flex-ER takes inspiration from these approaches, extending them to include experimental tasks.

2.2 Designing user studies for information visualization

Several researchers propose guidelines to design user studies for information visualization. For example, the nested model for information visualization suggests [44] that the evaluation depends on the abstraction level, and divides the visualization design and evaluation into four layers. Lam et al.,[36] present seven scenarios of different visualization evaluations. More recently, Elmqvist and Yi propose a series of patterns to better evaluate visualizations [19]. They classify a visualization evaluation as quantitative or qualitative. Quantitative studies use performance measures such as error and time. Among these patterns, some are specially adapted to controlled user studies, and thus work well as use cases for Flex-ER. For example, *Factor Mining* and *Trial Mining* address the problem of deciding which trials and factors to include in a user study. The pattern *Pilot Study* is also relevant as it consists of conducting multiple pilot studies to correctly choose some study's parameters.

Several other systems aim at supporting researchers in the creation and administration of user studies. For example, Englund et al., [20] introduce a system to design crowdsourced studies for scientific visualization. They also include data processing and presentation. Turton et al.,[61] present ETK a web-based set of modules to design and conduct visualization perception user studies. Evalbench [1] is a library to conduct visualization evaluations. Touchstone [41] is a system to help researchers design user studies in HCI. Touchstone2 [18] extends the concepts enabling users to test trade-offs when designing experiments. NexP [43] helps beginners design user studies for HCI. MRAT [47] enables experiment designers to collect and visualize data coming from interaction techniques developed in Unity, and test them using diverse tasks without coding. Another possible approach is the use of TestBeds, which have successfully been used for evaluating virtual reality selection and manipulation techniques [8, 52].

All of these tools ease designing or conducting user studies in specific scenarios. However, none of them target specifically immersive analytics interaction techniques. Flex-ER is built around components specific to IA studies, to facilitate manipulating and combining them.

Comparing Environments	Comparing Visualizations	Comparing Interaction Techniques
[5], [15], [64], [28], [24], [34], [33], [29], [63], [21], [22]	[65], [67], [64], [62], [35], [39], [34], [33], [66], [11]	[53], [66], [17], [60]

Table 1. Categorization of existing IA user studies.

2.3 Studies in Mixed Reality

There is a plethora of interaction techniques for mixed reality that have been subject to user studies. A comprehensive review of selection techniques can be found in Argelaguet et al. [4]. We focus here specifically on studies addressing immersive analytic applications.

We review existing user studies targeting immersive analytics and categorize them according to their goal: comparing environments, visualization or interaction techniques, as presented in the results of the study. We selected the papers based on keyword searches containing "user study", "empirical", "immersive visualization" and "immersive analytics". We only kept papers that provided some kind of interaction in the evaluated visualizations and that are less than 10 years old. This resulted in 23 papers listed in Table 1. Most of the reviewed papers either compare devices or visualization designs. The most frequent environments are Virtual Reality, present in 17 papers, and Desktop, present in 12 papers. When comparing environments the most frequent comparison is VR vs Desktop. When comparing visualization or interaction techniques, most studies consider only one environment. Furthermore, all these studies only consider one hardware and environment setup. For example, they only use one specific type of VR headset. Regarding interaction techniques, the most frequent interaction techniques are, according to Yi et al. [68] classification, for *selecting* and *exploring* the data. Only 8 studies propose interaction techniques to *reconfigure* and 4 to *abstract/elaborate* and *filter*.

3 CHALLENGES

Based on the literature review (Section 2) and our experience designing user studies, we address two challenges to ease the creating, testing and conducting of user studies in immersive analytics.

C1 - Replicability: The review of existing studies showcase an interest for comparing interaction techniques and visualizations across environments to assess the benefits and drawbacks of different devices. Nevertheless, most of these studies only consider one hardware and one software setup, or do not test all available environments. Furthermore, the code to conduct the experiment is rarely shared. In order to facilitate reproducing the results, it should be easy to share the experiment setting. Furthermore, to facilitate reusing user studies, for example, to test internal validity [23], it should be easy to adapt existing user studies to other environments, visualizations and interaction techniques, through a modular approach.

C2 - Task Oriented: The interaction techniques supported in the visualizations considered in the reviewed studies consider almost all common information visualization interaction techniques, classified by Yi et al. as select, explore, reconfigure, encode, abstract/elaborate, filter, and connect [68]. Nevertheless, the tasks used to compare these visualizations vary widely, from value retrieval to selection tasks. In order to support device comparability and generalization, the platform should support low level tasks [26] that can be used as baselines when comparing devices. Thus, the platform should be able to represent these low level tasks easily.

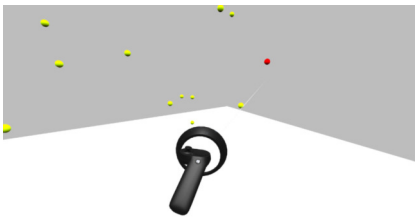


Fig. 2. Screen capture of the brushing task using VR. The points are distributed according to a random function with predefined values for the horizontal and vertical fields of view.

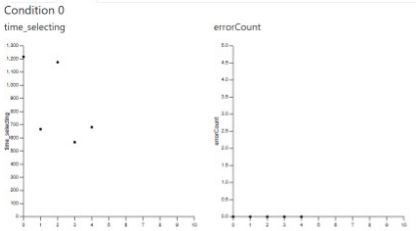


Fig. 3. Screenshot of the results preview in the Flex-ER web-based user interface, for the VR condition. The measures are defined in the JSON file.

4 FLEX-ER

4.1 Overview

Flex-ER relies on a JSON specification to describe experimental conditions. We use JSON because it has been successfully used to define grammars of graphics and immersive visualizations [58, 59]. We chose to encompass the whole setting definition in one file to facilitate sharing and reproducing user studies (C1). This specification file is based on a component-based structure to facilitate reusing elements. Based on how researchers describe IA user studies in the reviewed papers, we separate the setting in the data that will be visualized, the visualization of the data, the interaction techniques, the task, and the apparatus for each condition. This results in six main components: *data*, *inputs*, *devices*, *marks*, the interaction state machine *ISM*, and the task state machine *TSM*, as depicted in Figure 1. Finally, the *conditions* component, declares which devices, inputs and marks are used for each condition. The complete description of Flex-ER JSON specification can be found as supplemental material. In order to facilitate designing experiments that rely on Flex-ER JSON specification, we provide two user interfaces: a *Designer*, to test and debug experimental conditions using program visualization, and a *Runner*, to conduct the experiments.

4.2 Our Components

We use an example based on a task comparing brushing across three devices to illustrate Flex-ER components. Brushing is an interaction technique used to select source points in a view by direct manipulation [32]. This example uses a low level task to represent brushing in 3D point clouds and compares three devices that support different extended reality modalities: mobile augmented reality (Ipad), head mounted augmented reality (Hololens) and virtual reality (head mounted display). The rendering result of this task in the VR headset is depicted in Figure 2.

4.2.1 Data and marks. The *data* component encompasses both the definition of the data to be visualized and the visualization settings. In order to make Flex-ER as compatible with Vega Lite **[57]** as possible, data can come from a JSON file using the same format as Vega Lite. Furthermore, we provide some standard random data generation methods, with user defined constraints, to support testing data parameters such as the field of view [50]. The visualization is then defined through an array of *marks*. Each mark corresponds to a visualization of the data, mapping marks to visual channels to data attributes through an encoding following Vega Lite's model. However, to be able to reference the mark in other components of the experiment, the mark is declared with a unique ID in addition to the mark TYPE. Furthermore, to position the mark in 3D space, users can also declare its position, width, height and depth. For now,

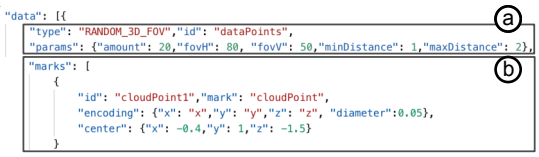


Fig. 4. JSON used to describe the data and marks for the brushing example.

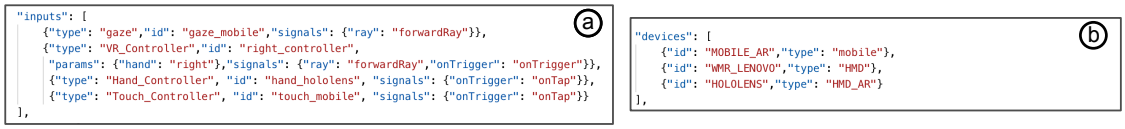


Fig. 5. JSON used to describe the inputs and devices for the brushing example.

Flex-ER supports bar charts and 3D scatterplots. It is also possible to define *marks* in an independent component, through its ID, TYPE and ENCODING. These marks can then be reused in the *data* component or in the state machines, using the mark ID in the MARK field.

For the brushing example, we use a 3D point cloud since it represents the use of 3D scatterplots in immersive analytics [5, 53], and it has been extensively used for testing pointing techniques in VR [40, 50]. We, therefore, define the data as a set of 3D points, using a predefined function, that generates 3D points in a constrained field of view, using the code depicted in Figure 5. This will generate 20 points, at a minimum distance of the center of 1 meter and a maximum distance of 2 meters, with 80 degrees as horizontal field of view and 50 degrees for the vertical field of view. We use the *mark* type point, where the x,y,z coordinates of each point will be mapped to the x,y,z generated values. The data will be located at (-0.4,1,1.5), as visible in Figure 5-(b). All points have the same diameter of 25 cm.

4.2.2 Devices, Inputs and Signals. Following C1, Flex-ER handles diverse extended reality tools, such as HMD and augmented reality devices, declared in the *devices* and *inputs* components. The *device* represents the physical tool in charge of displaying the visualization, such as a head mounted display or a mobile device, and is defined by an ID and a TYPE. Devices can be attached to *inputs*. We define an *input* as a signal emitter, that acts as an abstraction of the devices' events. For example, the same gaze input can receive events from a Hololens, a mobile phone and a VR headset as all of them rely on a 3D camera to navigate. Furthermore, an *input* is defined by its TYPE and ID. Each one emits a series of *signals*, according to its type. Every *input* emits a position signal, which is updated every time the input position changes. For now, the supported inputs are GAZE, for all devices, HANDCONTROLLER for the hand detected while using the Hololens, TOUCHCONTROLLER for touch events on mobile touch devices, and VRCONTROLLER.

In the JSON specification, the three possible devices are declared in the DEVICES section of the JSON, through their TYPE, and ID to be referenced afterwards.

Then, the *devices* have to be linked to the *inputs*. In this experiment, we chose the GAZE input for the Hololens and the Ipad, and the VRCONTROLLER input for the VR device. Both of these inputs emit a FORWARD RAY signal, used as the primary mechanism for target selection. We also define trigger signals for these inputs based on the trigger button on the VR controller, the selection gesture on the Hololens, and a tap on the Ipad screen. All of these inputs provide the same signals: RAY and ONTRIGGER that will be used to drive the interaction.




```
"states":[
   {   "id": "selecting",
       "actions": [
           {"id": "hit","signals": ["ray"],
            "apply": {"value": "pick",
                      "args": {"ray":"ray"}}
           },
           {"id":"selected","signals": ["onTrigger"],
            "apply": {"value": "hit.mesh"}}
       ],
       "marks": [
           {"id": "laserRight","mark": "laserRay",
            "encoding": {"length": "hit.distance","parent": "right_controller"}
           },
           {"id": "targetMesh","mark": "cursorTarget",
            "encoding": {"parent": "gaze_mobile","hit": "hit","ray": "ray"}
           }
       ]}],
```

```
"task": {
    "repetitions": [
        {"id": "random_repetition","amount": 2,"data": "dataPoints",
        "order": {"type": "random","repeat": "false"}
        }],
    "measures": [
        {"id": "time_selecting","state": "selecting_start","type": "ellapsedTime"},
        {"id": "errorCount","state": "selecting_start","value": "selecting_start.error",
        "type": "count"}
    ],
    "states": [
        {"id": "selecting_start",
         "actions": [
                {"id": "nextRandom","signals": ["enter"],
                 "apply": {
                    "value": "getNext",
                    "args": {"repetition": "random_repetition","objects": "cloudPoint1"}}},
                {"id": "nextTarget","signals": ["nextRandom"],
                 "apply": {
                    "value": "setAttribute",
                        "args": {"object": "nextRandom","attribute": "selected",
                        "value": "true"}}},
                {"id": "correct","signals": ["selecting.selected"],
                 "apply": {"value": "isEqual",
                    "args": {"target": "nextTarget.mesh","selected": "selecting.selected"}}},
                {"id": "error",
                 "signals": ["selecting.selected"],
                 "apply": {"value": "isDifferent",
                        "args": {"target": "nextTarget.mesh","selected": "selecting.selected"}}}],
            "transitions": [{"state": "selecting_start","condition": "correct"}],
            "marks":[
                {"id": "cloudPoint1","encoding": {
                        "color": {"selected?": "rgb(255,0,0)","value": "rgb(200,255,0)"}}
                }]
        }
    ]
}
```

Fig. 6. JSON used to describe the ISM and the TSM.

4.2.3 Interaction State Machine. Flex-ER uses a state machine model to define the interaction techniques. Our model is inspired by the FlowState model [2, 3] since it couples data flow with a state machine and is simple to grasp and use. The state machine is defined by a series of states, and stores the current state. Transitions enable the system to go from one state to another according to incoming events. However, we adapted the model to be able to calculate and store global variables inside each state. Thus, in the Flex-ER JSON specification, a state machine is defined by an array of *states*, where each state has an array of *actions* and *transitions*.

We define *actions* to be able to calculate and store values internal to each state. An *action* is defined by an id, and is calculated when receiving a *signal*, that is transformed using a pre-defined function. The result of this transformation will then be stored, and can then be used as a *signal* for other *actions* or for dispatching a *transition* between states. Each state *transition* is defined by a *condition* based on a boolean *signal* and the *state* that will be set as current if the *condition* is granted. Furthermore, states can define *marks* and their *encodings*. Aside from being used for the visualization, *marks* are also used for input feedback, such as a laser ray for VR controllers, using 3D geometric primitives such as cylinders, spheres, and rings. For now, our model is limited to one state machine to handle interactions.

In our example, there is only one state, SELECTING, as the behavior of the selection technique does not change. The SELECTING state is defined by two actions, HIT and SELECTED (Figure 6-(a)). HIT is updated every time that the signal RAY changes by applying the function PICK to the data objects, that will return the closest intersection between them and the RAY. SELECTED is used to store the hit object once the trigger is pressed. This state also defines marks for the interaction objects: LASERRIGHT and TARGETMESH. Each of them has a parent, that has to match one of the inputs, and an encoding. The encoding can be used to give a visual cue that highlights the object that will be selected. For example, the LASERRIGHT mark length will be updated so that the ray stops at the closest intersected object.

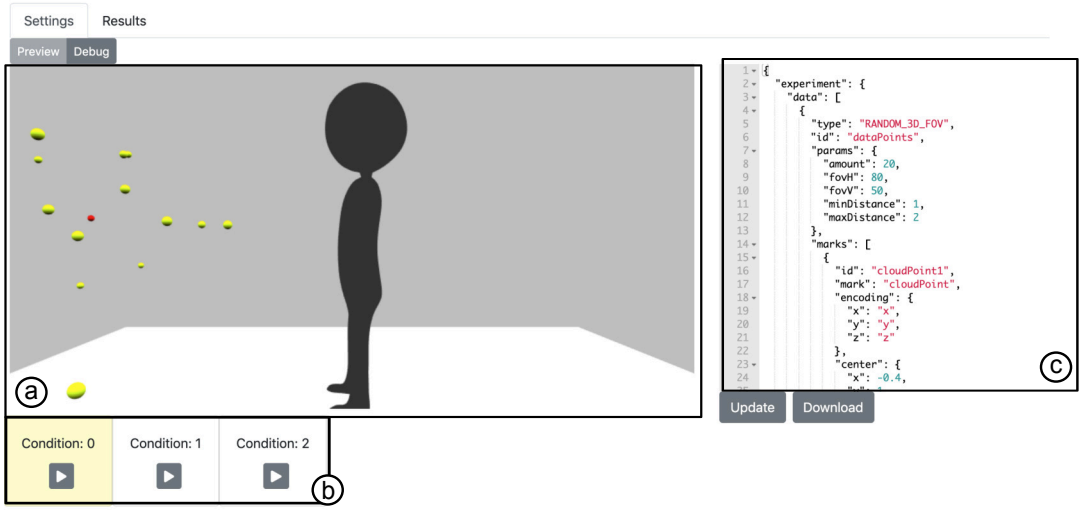


Fig. 7. Screenshot of Flex-ER web-based user interface. The condition preview (a) displays the viewer position and a preview of the visualization. The user can edit the experiment setting in the JSON editor (c) to see the changes in the preview. He can choose which condition to play by selecting it on the condition bar (b).

4.2.4 Task State Machine and Measures. Multiple tasks have been used in studies evaluating immersive visualizations, including distance comparison [67], finding outliers [53] and flow magnitude comparison [65]. These tasks can be simple, like selecting one target after the other, or more complex, such as revealing the target object only at a certain distance from it. Therefore, to support a wide range of tasks, we use a second state machine, the TSM, that works exactly like the ISM, to handle task *states*.

Besides the TSM *states*, the *task* component also includes the *measures* and *repetitions* definition. *Measures* are used to evaluate task performance, and record participants' behavior. Flex-ER enables users to define *measures* derived from *states* and *actions*. For example, the time spent in each state is recorded by default in the measure type ELLAPSEDTIME. It is also possible to apply some generic functions to specific actions to use them as measures, such as COUNT, SUM or MEAN. *Repetitions* are used to define consecutive trials, that iterate through data items, for example to repeat the same task with different random targets.

In this case, the task state machine has one state, SELECTINGSTART that defines four actions (Figure 6-(b)). The action NEXTRANDOM will be defined by the GETNEXT function that returns an object of the specified MARK according to an REPETITION. The NEXTTARGET action will set the selected attribute value to true in the object stores in NEXTRANDOM. The CORRECT action verifies if the selected target action (calculated in the ISM) matches the target, and the ERROR action verifies the contrary. A transition verifies if CORRECT is true, in that case, the state machine re-enters the state, triggering the selection of a new target. The state also handles the encoding of the target object. For this, a conditional encoding is set on the color, depending on whether the object has its SELECTED attribute set to true or not.

The time is measured for each target acquisition through the TIMESELECTING measure, by measuring the time spent on the task state, and the number of errors through the ERRORCOUNT measure, that counts how many times an action is triggered, in this case, the ERROR action.

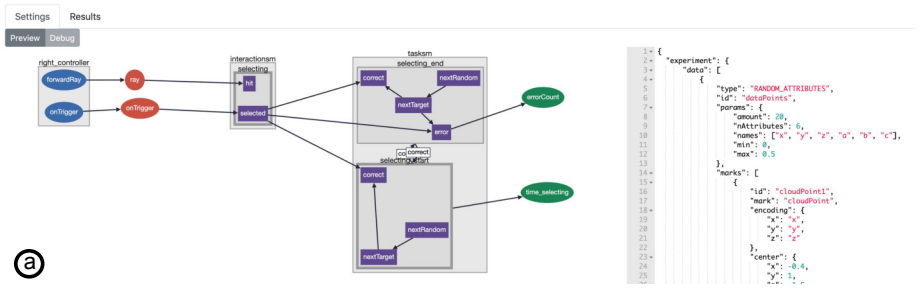


Fig. 8. (a) State machine visualizer for a linking task. Inputs are depicted in blue, input signals in red, actions in purple and measures in green. Arrows are used to depict dependencies, for actions, arrows represent when a value is being used as a signal, for measures, they represent the origin of the value they are capturing.

4.3 User Interface

In order to facilitate the design of the experimental conditions and interaction techniques, we propose two web-based user interfaces: the *Designer* and the *Runner* UI.

4.3.1 The Designer UI. The first user interface aims at facilitating the experimental condition's design and prototyping, depicted in Figure 7.As depicted in Figure 7-a, the user can edit, thanks to the *Preview* tab, the JSON and visualize the resulting *visualization* and their position according to the viewer. To ease the JSON specification, we provide a visualization of the ISM and the TSM on the *Debug* tab. The visualization depicts the inputs, the signals, the states and the actions of both state machines, and their links to the code editor. Once the user selects an item in the state machine visualization, the code that defines it is highlighted, as depicted in Figure 11 and in the companion video.

Because actions relate differently between them, providing all the information in the state machine diagram would have resulted in an occluded view, as might happen with visual programming languages [27]. Thus, we rather chose to restrict all editing to the JSON code, and use the state machine visualization as a program visualization to help debugging [46].

The user can also select a condition and play it to test it. To further facilitate debugging, the current states of the running condition will be updated in the state machines visualization. To preview the results of a particular condition, the measures collected during the test will be displayed as the user realises the experiment in the *Results* tab. Each measure is plotted once every time it is recorded, and the results can be downloaded as a CSV file. A screenshot of the results preview is depicted in Figure 3.

4.3.2 The Runner UI. The second user interface enables experiment operators to run the experiments described with the Flex-ER specification. A second JSON file is used to specify the blocks and trials for a participant. For now, each participant sequence can be divided in a set of blocks and each block can contain multiple trials. Each trial references one of the conditions defined in the file describing the experiment and can be repeated a specified number of times.

The runner interface will indicate the progress on the current study, which device is being used at the current trial and the URL address of the condition. To actually run the condition, the participant opens the indicated URL on the navigator on the corresponding device and runs it by pressing the "Start Trial" button. Furthermore, the runner window enables the experiment runner to download the results at any time.

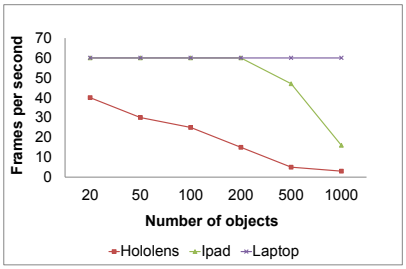


Fig. 9. Performance evaluation for the Hololens 1, an Ipad Pro and a Macbook Pro.

4.4 Implementation

The user interfaces of Flex-Er are programmed using React[1]. The 3D visualizations are based on the libraries BabylonJS[2] and ThreeJS[3], that work on top of WebVR[4], WebXR[5] and WebGL [6]. The state machine visualizer uses Cytoscape [7]. The system uses a client-server architecture. The clients hosts both the designer and the runner. The communication between the clients and the server is based on web sockets. Every time the setting is modified, all the other clients are notified.

Once the user runs the JSON script, it is parsed and then used to create both the visual representation using BabylonJs or ThreeJS primitives, while handling the events. The system is based on the use of Babylon Observables (i.e. observable design patterns). First, we create observables for all marks, inputs, signals and actions. Then, we transform this description into visual objects and add the event listeners according to the signal and action descriptions. Once the current interaction handler state or the task state changes, the events listeners are detached and reattached according to the new state specification.

Flex-ER web user interfaces can be used as any React Component. All the functions in charge of attaching the input, the listeners, of creating the marks, and the apply functions are stored in Javascript maps. To include custom functions, users can store them in the global map variables customApplyFunctions, customInputListeners and customMarkBuilders according to the API specifications before importing the Flex-ER component, so they will be added to the predefined functions. For example, the customApplyFunctions variable maps the apply functions names to the javascript functions. All the values declared in the args object of the JSON file will be accessible in the custom function, as the BabylonJS or ThreeJS scenes.

Presently, Flex-ER works on any internet browser supporting WebVR. For mobile, we use the Mozilla XRViewer[8] on IOS because BabylonJS does not yet handle WebXR with augmented reality. We evaluated Flex-ER performance using a Hololens 1, an Ipad Pro second series, and a MacBookPro Retina 15" equipped with an NVIDIA GeForce GT 750M 2048MB. The results for rendering the 3D scatterplot described in Section 5-2 are depicted in Figure 9.

[1]https://reactjs.org
[2]https://www.babylonjs.com
[3]https://threejs.org
[4]https://webvr.info
[5]https://developer.mozilla.org/en-US/docs/Web/API/WebXR$_{D}evice_{A}PI$
[6]http://khronos.org/webgl
[7]http://js.cytoscape.org
[8]https://labs.mozilla.org/projects/webxr-viewer/

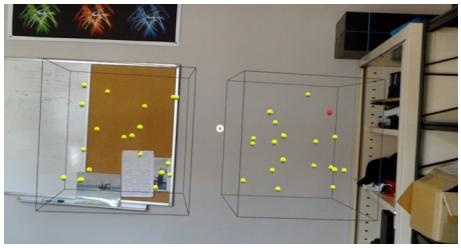


Fig. 10. The two views used for a linking task in the Hololens.

5 EXAMPLES

This section reports our evaluation by use-cases, inspired by Ledo et al. [37]. As such, we present some examples, where we showcase the expressiveness of the JSON specification through two low-level task examples (C2), and the usefulness of the web based interface to facilitate debugging and previewing experiment conditions (C1). In the following, we provide the high-level description of the state machines describing these examples while the full JSON definitions are given in the supplemental material.

5.1 Debugging Linking Task

Following the second challenge (tasks), Flex-ER must support basic information visualization tasks such as brushing, navigating and linking. Brushing, i.e. example described in Section 4, consists of the selection, by direct manipulation, of source points in a view [32]. Brushing will result in a visual representation of the relationships between the source points and the target points in coordinated views, called linking [32]. This scenario corresponds to a use case where a researcher wants to extend the custom brushing technique with linking, and compare it across the three devices compared to in the Section 4.

To design a low-level task to represent linking, the researcher considers that the user, when exploring linked views, goes from one view to another to inspect the different attributes of a selected object. Therefore, they design a pointing task where the user has to select first an element in a view, and then inspect the related element in a second view. They first decide to test the potential color encoding. Since Flex-ER specification separates the task from the interaction management, the researcher only needs to modify the data and task definitions of the brushing setting. The resulting state machines for this task are depicted in Figure 8.

Similar to the previous experimental design, the researcher uses random data. However, this time they generate 3D points inside specific ranges and with six numeric random attributes: x,y,z,a,b and c. This experiment contains two views, as depicted in Figure 10 representing different attributes of the same data. They keep the same visualization than before, but for this task they include an explicit separation of the two views, by rendering a border around the points. Each visualization encodes three of the six attributes of the data through the points' position. The apparatus and the interaction technique are the same as the ones defined for the brushing experiment in Section 4.

This task, contrary to the brushing task, needs two states to handle the two-step selection: a first state, called SELECTING_START handles the target selection in the source view, and a second state, called SELECTING_END, handles it in the target view. The SELECTING_START state is similar to the task state of the brushing experiment, except that there are now two marks instead of one, ie. one for each view.

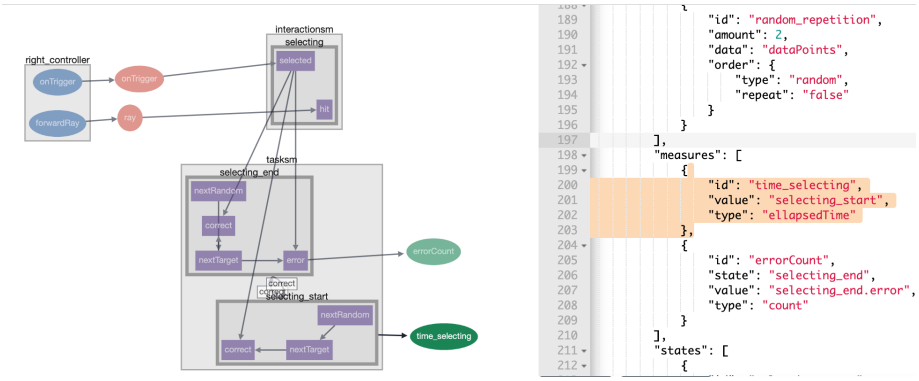


Fig. 11. Once the user clicks on a state machine node, the code corresponding to the node is highlighted. In this case the user clicks on the time measure node to correct the wrong state.

The cloud point encoding depends on the state; if the current state is SELECTING_START, the selected target will have a different color in the source view, if it is SELECTING_END, the selected target will be different on the target view. The *action* used to compare the current selection to the target element will be calculated in the SELECTING_END state, as the measures, as this state corresponds to the user linking the source view to the target view.

Once the researcher has finished the JSON setup, they test the condition using an iPad by pressing the play button on the corresponding condition. Looking at the *results* tab at the same time than they conduct the experiment, they notice that the time measure is being recorded before than the error measure. To address this issue, they open the *debug* tab, and notice that the time measure is attached to the SELECTING_START state instead of the SELECTING_END state. By clicking on the node representing the measure, the corresponding code is highlighted, as depicted in Figure 11, and they can correct it by replacing the state SELECTING_END in the code. This visual mapping enables users to rapidly see the relationship between the task elements, thus facilitates debugging the interaction techniques and the task inner working.

5.2 Adapting a navigation task

For our second scenario, we imagine that a researcher wants to adapt a user study he received from a colleague. The original user study compares two AR devices, an iPad and an Hololens. This experimentation uses an interaction technique for the tasks of panning and zooming, i.e. explore and elaborate [68] the scene, and uses a 3D scatterplot as in the previous examples (see companion video). The low-level task used in this case is again selecting the element of interest, except this time the object is visible only when the *visualization* is close enough to the participant. This task is commonly used for comparing multi-scale navigation techniques [51].

This task is defined through a state machine with three states in Flex-ER. The first one, INIT initialises the position of the *visualization*, so that the it is at the same position every time a trial begins. Once the mark position is setup, there is a transition to the TARGET INVISIBLE state, where an action is used to calculate the distance between the camera and the mark. If the distance is less than a specified threshold, the state machine enters the TARGET VISIBLE state where the target is red instead of yellow, and the user can select it to begin the next trail.

The navigation is achieved in both devices through panning and zooming. Both devices receive the signals PAN. The panning is controlled by the hand position given by the Hololens, and with a drag gesture on the iPad. Because the

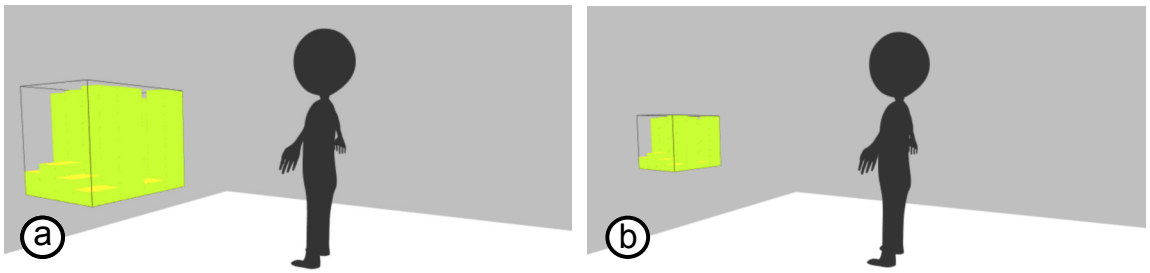


Fig. 12. Preview visualizations for different bar chart sizes (a) size of 1 meter per side, (2) size of 0.5 per side.

panning in the iPad only considers two dimensions, the PINCH signal is used to move the MARK along the direction of the ray between the camera and the MARK, to zoom-in or zoom-out.

In this use case, the researcher would like to change the visualization, and use a bar chart instead. To achieve this, they change the data definition in the JSON specification to use data coming from a JSON file instead, and to encode it through a bar mark, keeping the same id attribute for the data and the mark that was used in the original setting. They then add the bar mark definition to the marks section of the file. Once they update the designer setting, they realise through the visualization previewer that the bar chart is too big in relation to the viewer size, as depicted in Figure 12-a. To correct this, they try different mark sizes by changing the corresponding attribute in the code, until they are satisfied with their choice, depicted in Figure 12-b. The task will work exactly as before: the bar chart will be positioned at the initial position at the beginning of the trial and the target bar will be displayed only once the viewer is close enough to the mark.

6 USER EVALUATION

In order to assess Flex-ER, we collected qualitative feedback, following a walkthrough demonstration approach [37]. Walkthrough demonstrations can help collect feedback on the utility of a toolkit, as the participant does not directly use the toolkit, they can focus on the value of having access to it. We conducted interviews with three researchers (1 female, 2 male, ages from 28 to 45) and two research engineers (2 males, ages from 23 and 37) working on immersive visualizations.

6.1 Procedure

We conducted remote video interviews that lasted 40 minutes in average. We started by collecting information about the participant background, his or her common practices regarding immersive visualizations and user studies in general. We then presented Flex-ER, explaining first the JSON specification and second, demonstrating the system through a demo with the two uses cases previously described in Section 5. We encouraged the participants to regularly ask questions if needed. The interview ended with a series of questions to collect their feedback.

6.2 Participant background and common practices

The five participants had already designed immersive visualizations for different applications domains: aeronautics (P2, P3), geovisualization (P4,P5) and energy visualization (P1). Three of them had already designed, conducted and implemented user studies to evaluate immersive visualization interaction techniques: P3, P4 and P5 rely mostly on web based technologies, P1 on Unity, and P2 on C. P1 is the only one who used another tool for prototyping, Maquette, and

was also disappointed by the tool due to its lack of flexibility. All of them mentioned using different devices, such as Hololens, VR Headsets and mobile augmented reality. However, they mentioned that they mostly develop applications with only one device in mind. Nevertheless, P3 had to develop the same interactive visualization for desktop and for VR headeset, while P1 for Hololens and VR headsets. P2, working with physical displays, switches between a desktop simulator and the actual physical display. The three of them mentioned that some code changes are required to switch between these devices, making it cumbersome to switch between devices while testing.

Regarding their code sharing practices, two of them share their application code on public repositories, but none of them share the code used to run the experiments. P1 mentioned that this code is too specific to each project. P1 and P2 also share systematically their experiment results and statistical code to process the results.

6.3 Flex-ER qualitative feedback

Overall, the participant feedback was positive. When asking them if they though the tool could be useful for them, P1 mentioned it could be especially useful to "generate very quickly studies", P2 mentioned that it could be helpful "when people are confined at home" to run experiments remotely, and P3, P4 and P5 stated that it would help testing different studies in different devices easily. P4 also mentioned that even if he does not usually test his applications with users he would be curious to do so using Flex-ER. P5 found the modularity of the approach particularly useful, as it enables reusing experiments, and fixing one aspect of the user study, such as the data, while varying another, such as the devices.

We then asked the participants to describe concrete use cases where Flex-ER could be helpful for their current research activities. We summarize two of them next.

6.3.1 Simulating a physical display. P2 is currently doing research on levitating particle displays [55]. To enable interaction techniques with these particles, he is studying different techniques to avoid collisions between them acting on one particle's trajectory through user interaction. He mentioned that Flex-ER could be useful to visualize these different techniques, as it would enable describing the different avoiding collision strategies and visualize the resulting 3D trajectories using 3D spheres instead of the actual physical particles. To achieve this, he could extend the system with the actions that receive the user input and calculate the trajectory of the selected sphere according to the strategy, that would determine the sphere position. He could then drive the interaction using an augmented reality display like the Hololens, for example.

6.3.2 Exploring interaction techniques. P5 studies how to facilitate urban planning tasks through geo-visualization. In this scenario, buildings can be visualized in an immersive environment using various attributes such as shape, volume and graphic representation. To simulate urban changes, it can be useful to interactively change the city, for example replacing a building with a different one. To achieve this, geo-visualization experts aim to design simple interactions that support easily navigating between different scales. This high level task could potentially be represented with a more low level task, in order to compare different navigation techniques. Furthermore, using Flex-ER, different interaction techniques and tasks could be quickly tested and showcased to final users as to observe and learn about their practices, to improve their design.

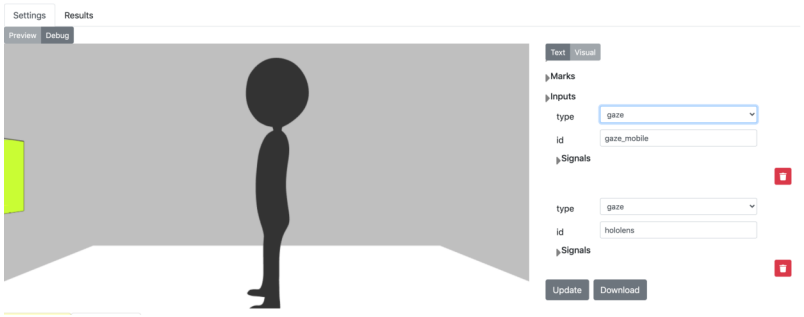


Fig. 13. Form based GUI to facilitate the JSON editing. Users can choose predefined values from fixed lists of values, to avoid mistakes.

6.4 Improvement possibilities

At the end of the interview we asked participants if there were some other features they would like to see in the environment and if they would change something about it. We summarize here three possibilities for improvement, mentioned by several participants.

6.4.1 JSON specification. P1, P3, P4 and P5 mentioned that it might be difficult to navigate and understand the JSON code. P1 suggested that, even if having the whole study description in one file is useful, providing a visual encoding of the different components might facilitate navigating in the code. P3 would like to have more feedback about what the predefined function do in the interface. P4 mentioned that having some predefined templates in the interface would make it easier to test different experiments. P5 would like to be able to edit the code through the state machine visualization, through direct manipulation of nodes and edges. As stated in Section 4, we chose to limit the editing to the JSON file, to avoid occlusion. Nevertheless, to partially alleviate this problem, we implemented a GUI based on the JSON structure, that better depicts the different elements in the definition and constraints the inputs to limit errors, as depicted in Figure 13.

6.4.2 Importing 3D objects. P1, P3 and P5 noted that it could be useful to import 3D objects in the scene, both to use them as visual marks and to have more realistic scene backgrounds in the VR conditions. P1 mentioned that using VR can be useful to simulate environments that are not accessible otherwise, and P5 said it might be interesting to study the background effects on the visualizations. Adding these functionalities to Flex-ER is straightforward because Three.js and Babylon.JS support importing 3D objects.

6.4.3 Replaying conditions. P1, P4 and P5 would like to be able to replay the conditions after they have ended, for example replaying the interaction or the positions of the participants. Because we use observables to handle all input signals and actions, we can also use them as measures in the task state machine, and log them. Therefore, adding a replay function would only require adding an additional visualization to plot the different values across time.

7 DISCUSSION

We presented Flex-ER, an environment to prototype experimental conditions for Immersive Analytics (IA) user studies. Flex-ER takes inspiration from existing visualization toolkits and grammars such as Vega [57], IATK [14] and DXR [59], but also includes experimental tasks definition. We discuss Flex-ER limitations and future work.

7.1 Evaluation

We evaluated Flex-ER using a demonstration and a walk-through approach [37]. We chose two use case examples based on existing abstract tasks to evaluate interaction techniques: select, connect, abstract/elaborate and explore, and conducted remote interviews with five users working in immersive analytics. However, our evaluation is limited by the number of examples (2) and the number of participants (5). Even if we did not provide examples for the other three information visualization interaction techniques identified by Yi et al. [68] we believe the task definition is flexible enough to describe them. For example, the same manipulation techniques used for the navigation task described in Section 5 could be used to test interaction techniques to reconfigure the views.

Furthermore, the participants did not use the system themselves. We chose this approach to let them focus on the potential usability of the system rather than on learning how to use it. Thus, it is hard to assess the threshold [45] of the system at this stage. Nevertheless, the participants' feedback suggests that the JSON specification might be hard to understand. We believe that the GUI added after the evaluation might partially alleviate this problem. Further evaluations, for example, an observation study [37] where researchers can use the system themselves to implement studies, would give us further insights about the usability of Flex-ER.

7.2 Flexibility

We chose to use a simple model for the interaction handling, allowing only one state machine. We also chose to use this approach because of its simplicity, providing a low threshold [45], at the cost of lowering the ceiling [45]. This might limit the interaction techniques that we can represent. For example, it would be cumbersome to represent interaction techniques that require hierarchical state machines [6] such as different dragging behaviors according to a button press. Furthermore, Flex-ER is also limited by the existing pre-defined functions that create visualizations and handle input and interaction techniques. Nevertheless, users are able to add custom functions, but this requires coding them directly in Javascript.

Second, it is not possible to handle more than one device in a condition. It could be interesting to couple multiple devices with different input modalities, and it would be possible by extending the current client/server architecture. For example, to use a mobile phone as a joystick to navigate in the scene coupled with an Hololens for the visualization.

7.3 Scalability

We chose to implement Flex-ER using web technologies to ensure interoperability across devices. In addition, we focused on abstract tasks for testing interaction techniques. For these reasons, and because the rendering is not fully optimized, as depicted in Figure 9, visualizations with a large number of objects do not maintain a 60 fps rate in all devices. Using progressive visualization techniques [38], or more scalable rendering techniques [14, 54] could make Flex-ER more scalable, which we leave for future work.

8 CONCLUSION

We presented Flex-ER, a platform to prototype and conduct user studies to evaluate interaction techniques for immersive visualizations, aimed at making these kind of studies more reproducible. Flex-ER relies on a JSON specification that describes the data, the visualization, the interaction techniques, and the devices and inputs for a user study. The

interaction techniques are defined using a state machine, with enough flexibility to handle user defined input signals. These are independent of the device and can be easily reused. Furthermore, we provide a second state machine to handle the tasks, as to make the studies *task oriented*, and enable the reuse of these tasks. We presented two use cases where we showcased how Flex-ER supports describing two low-level tasks to evaluate four of the seven information visualization interaction techniques identified by Yi et al. [68]: select, connect, abstract/elaborate and explore. Flex-ER was well received by five users working on immersive visualizations in a preliminary user evaluation, as they appreciated its modular approach. Furthermore, their qualitative feedback provides interesting directions for future work such as adding the possibility to import 3D objects and replaying conditions.

ACKNOWLEDGMENTS

This research was partly supported by SESAR H2020 project ENVISION grant agreement No 783270. We thank Guillaume Truong for his technical help.

REFERENCES

[1] W. Aigner, S. Hoffmann, and A. Rind. 2013. EvalBench: A Software Library for Visualization Evaluation. *Computer Graphics Forum* 32, 3pt1 (2013), 41–50. https://doi.org/10.1111/cgf.12091 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12091

[2] Caroline Appert and Michel Beaudouin-Lafon. 2006. SwingStates: Adding State Machines to the Swing Toolkit. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology* (Montreux, Switzerland) *(UIST '06)*. Association for Computing Machinery, New York, NY, USA, 319–322. https://doi.org/10.1145/1166253.1166302

[3] Caroline Appert, Stéphane Huot, Pierre Dragicevic, and Michel Beaudouin-Lafon. 2009. FlowStates: Prototypage D'Applications Interactives Avec Des Flots De DonnÉEs Et Des Machines À ÉTats. In *Proceedings of the 21st International Conference on Association Francophone D'Interaction Homme-Machine* (Grenoble, France) *(IHM '09)*. ACM, New York, NY, USA, 119–128. https://doi.org/10.1145/1629826.1629845

[4] Ferran Argelaguet and Carlos Andujar. 2013. A survey of 3D object selection techniques for virtual environments. *Computers Graphics* 37, 3 (2013), 121 – 136. https://doi.org/10.1016/j.cag.2012.12.003

[5] B. Bach, R. Sicat, J. Beyer, M. Cordeil, and H. Pfister. 2018. The Hologram in My Hand: How Effective is Interactive Exploration of 3D Visualizations in Immersive Tangible Augmented Reality? *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan 2018), 457–467. https://doi.org/10.1109/TVCG.2017.2745941

[6] Renaud Blanch and Michel Beaudouin-Lafon. 2006. Programming Rich Interactions Using the Hierarchical State Machine Toolkit. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (Venezia, Italy) *(AVI '06)*. ACM, New York, NY, USA, 51–58. https://doi.org/10.1145/1133265.1133275

[7] M. Bostock, V. Ogievetsky, and J. Heer. 2011. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (Dec 2011), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

[8] Doug A. Bowman, Donald B. Johnson, and Larry F. Hodges. 2001. Testbed Evaluation of Virtual Environment Interaction Techniques. *Presence: Teleoperators and Virtual Environments* 10, 1 (2001), 75–95. https://doi.org/10.1162/105474601750182333 arXiv:https://doi.org/10.1162/105474601750182333

[9] W. Broll, I. Lindt, J. Ohlenburg, I. Herbst, M. Wittkamper, and T. Novotny. 2005. An infrastructure for realizing custom-tailored augmented reality user interfaces. *IEEE Transactions on Visualization and Computer Graphics* 11, 6 (Nov 2005), 722–733. https://doi.org/10.1109/TVCG.2005.90

[10] P. W. S. Butcher, N. W. John, and P. D. Ritsos. 2020. VRIA: A Web-based Framework for Creating Immersive Analytics Experiences. *IEEE Transactions on Visualization and Computer Graphics* (2020), 1–1.

[11] W. Büschel, S. Vogt, and R. Dachselt. 2019. Augmented Reality Graph Visualizations. *IEEE Computer Graphics and Applications* 39, 3 (2019), 29–40.

[12] Zhutian Chen, Wai Tong, Qianwen Wang, Benjamin Bach, and Huamin Qu. 2020. Augmenting Static Visualizations with PapARVis Designer. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376436

[13] J. Choi, D. G. Park, Y. L. Wong, E. Fisher, and N. Elmqvist. 2015. VisDock: A Toolkit for Cross-Cutting Interactions in Visualization. *IEEE Transactions on Visualization and Computer Graphics* 21, 9 (Sep. 2015), 1087–1100. https://doi.org/10.1109/TVCG.2015.2414454

[14] M. Cordeil, A. Cunningham, B. Bach, C. Hurter, B. H. Thomas, K. Marriott, and T. Dwyer. 2019. IATK: An Immersive Analytics Toolkit. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 200–209. https://doi.org/10.1109/VR.2019.8797978

[15] M. Cordeil, T. Dwyer, K. Klein, B. Laha, K. Marriott, and B. H. Thomas. 2017. Immersive Collaborative Analysis of Network Connectivity: CAVE-style or Head-Mounted Display? *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan 2017), 441–450. https://doi.org/10.1109/TVCG.2016.2599107

[16] Sarah H. Creem-Regehr, Jeanine K. Stefanucci, William B. Thompson, Nathan Nash, and Michael McCardell. 2015. Egocentric Distance Perception in the Oculus Rift (DK2). In *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception* (Tbingen, Germany) *(SAP '15)*. ACM, New York, NY, USA, 47–50. https://doi.org/10.1145/2804408.2804422

[17] A. Drogemuller, A. Cunningham, J. Walsh, M. Cordeil, W. Ross, and B. Thomas. 2018. Evaluating Navigation Techniques for 3D Graph Visualizations in Virtual Reality. In *2018 International Symposium on Big Data Visual and Immersive Analytics (BDVA)*. 1–10.

[18] Alexander Eiselmayer, Chat Wacharamanotham, Michel Beaudouin-Lafon, and Wendy E. Mackay. 2019. Touchstone2: An Interactive Environment for Exploring Trade-offs in HCI Experiment Design. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 217, 11 pages. https://doi.org/10.1145/3290605.3300447

[19] Niklas Elmqvist and Ji Soo Yi. 2015. Patterns for visualization evaluation. *Information Visualization* 14, 3 (2015), 250–269. https://doi.org/10.1177/1473871613513228 arXiv:https://doi.org/10.1177/1473871613513228

[20] R. Englund, S. Kottravel, and T. Ropinski. 2016. A crowdsourcing system for integrated and reproducible evaluation in scientific visualization. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*. 40–47. https://doi.org/10.1109/PACIFICVIS.2016.7465249

[21] Ugo Erra, Delfina Malandrino, and Luca Pepe. 2019. Virtual Reality Interfaces for Interacting with Three-Dimensional Graphs. *International Journal of Human–Computer Interaction* 35, 1 (2019), 75–88. https://doi.org/10.1080/10447318.2018.1429061

[22] R. Etemadpour, E. Monson, and L. Linsen. 2013. The Effect of Stereoscopic Immersive Environments on Projection-Based Multi-dimensional Data Visualization. In *2013 17th International Conference on Information Visualisation*. 389–397.

[23] Fiona Fidler and John Wilcox. 2018. Reproducibility of Scientific Results. In *The Stanford Encyclopedia of Philosophy* (winter 2018 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University.

[24] J. A. W. Filho, W. Stuerzlinger, and L. Nedel. 2020. Evaluating an Immersive Space-Time Cube Geovisualization for Intuitive Trajectory Data Exploration. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 514–524.

[25] Daniel Filonik, Tomasz Bednarz, Markus Rittenbruch, and Marcus Foth. 2016. Glance: Generalized Geometric Primitives and Transformations for Information Visualization in AR/VR Environments. In *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1* (Zhuhai, China) *(VRCAI '16)*. ACM, New York, NY, USA, 461–468. https://doi.org/10.1145/3013971.3014006

[26] A. Fonnet and Y. Prié. 2019. Survey of Immersive Analytics. *IEEE Transactions on Visualization and Computer Graphics* (2019), 1–1.

[27] GW French, J Richard Kennaway, and AM Day. 2014. Programs as visual, interactive documents. *Software: Practice and Experience* 44, 8 (2014), 911–930.

[28] N. Greffard, F. Picarougne, and P. Kuntz. 2014. Beyond the classical monoscopic 3D in graph analytics: An experimental study of the impact of stereoscopy. In *2014 IEEE VIS International Workshop on 3DVis (3DVis)*. 19–24.

[29] B. Hoppenstedt, T. Probst, M. Reichert, W. Schlee, K. Kammerer, M. Spiliopoulou, J. Schobel, M. Winter, A. Felnhofer, O. D. Kothgassner, and R. Pryss. 2019. Applicability of Immersive Analytics in Mixed Reality: Usability Study. *IEEE Access* 7 (2019), 71921–71932.

[30] C. Hurter, N. H. Riche, S. M. Drucker, M. Cordeil, R. Alligier, and R. Vuillemot. 2019. FiberClay: Sculpting Three Dimensional Trajectories to Reveal Structural Insights. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan 2019), 704–714. https://doi.org/10.1109/TVCG.2018.2865191

[31] N. W. Kim, E. Schweickart, Z. Liu, M. Dontcheva, W. Li, J. Popovic, and H. Pfister. 2017. Data-Driven Guides: Supporting Expressive Design for Information Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan 2017), 491–500. https://doi.org/10.1109/TVCG.2016.2598620

[32] P. Koytek, C. Perin, J. Vermeulen, E. André, and S. Carpendale. 2018. MyBrush: Brushing and Linking with Personal Agency. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan 2018), 605–615. https://doi.org/10.1109/TVCG.2017.2743859

[33] Matthias Kraus, Katrin Angerbauer, Juri Buchmüller, Daniel Schweitzer, Daniel A. Keim, Michael Sedlmair, and Johannes Fuchs. 2020. Assessing 2D and 3D Heatmaps for Comparative Analysis: An Empirical Study. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376675

[34] M. Kraus, N. Weiler, D. Oelke, J. Kehrer, D. A. Keim, and J. Fuchs. 2020. The Impact of Immersion on Cluster Identification Tasks. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 525–535.

[35] O. Kwon, C. Muelder, K. Lee, and K. Ma. 2016. A Study of Layout, Rendering, and Interaction Methods for Immersive Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics* 22, 7 (July 2016), 1802–1815. https://doi.org/10.1109/TVCG.2016.2520921

[36] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. 2012. Empirical Studies in Information Visualization: Seven Scenarios. *IEEE Transactions on Visualization and Computer Graphics* 18, 9 (Sep. 2012), 1520–1536. https://doi.org/10.1109/TVCG.2011.279

[37] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 36, 17 pages. https://doi.org/10.1145/3173574.3173610

[38] J. K. Li and K. Ma. 2020. P5: Portable Progressive Parallel Processing Pipelines for Interactive Data Analysis and Visualization. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 1151–1160.

[39] J. Liu, A. Prouzeau, B. Ens, and T. Dwyer. 2020. Design and Evaluation of Interactive Small Multiples Data Visualisation in Immersive Spaces. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 588–597.

[40] P. Lubos, G. Bruder, and F. Steinicke. 2014. Analysis of direct selection in head-mounted display environments. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*. 11–18. https://doi.org/10.1109/3DUI.2014.6798834

[41] Wendy E. Mackay, Caroline Appert, Michel Beaudouin-Lafon, Olivier Chapuis, Yangzhou Du, Jean-Daniel Fekete, and Yves Guiard. 2007. Touchstone: Exploratory Design of Experiments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. ACM, New York, NY, USA, 1425–1434. https://doi.org/10.1145/1240624.1240840

[42] Kim Marriott, Jian Chen, Marcel Hlawatsch, Takayuki Itoh, Miguel A. Nacenta, Guido Reina, and Wolfgang Stuerzlinger. 2018. *Just 5 Questions: Toward a Design Framework for Immersive Analytics*. Springer International Publishing, Cham, 259–288. https://doi.org/10.1007/978-3-030-01388-2_9

[43] Xiaojun Meng, Pin Sym Foong, Simon Perrault, and Shengdong Zhao. 2017. NexP: A Beginner Friendly Toolkit for Designing and Conducting Controlled Experiments. In *Human-Computer Interaction – INTERACT 2017*, Regina Bernhaupt, Girish Dalvi, Anirudha Joshi, Devanuj K. Balkrishan, Jacki O'Neill, and Marco Winckler (Eds.). Springer International Publishing, Cham, 132–141.

[44] T. Munzner. 2009. A Nested Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (Nov 2009), 921–928. https://doi.org/10.1109/TVCG.2009.111

[45] Brad Myers, Scott E Hudson, and Randy Pausch. 2000. Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)* 7, 1 (2000), 3–28.

[46] Brad A Myers. 1986. Visual programming, programming by example, and program visualization: a taxonomy. *ACM sigchi bulletin* 17, 4 (1986), 59–66.

[47] Michael Nebeling, Maximilian Speicher, Xizi Wang, Shwetha Rajaram, Brian D. Hall, Zijian Xie, Alexander R. E. Raistrick, Michelle Aebersold, Edward G. Happ, Jiayin Wang, Yanan Sun, Lotus Zhang, Leah E. Ramsier, and Rhea Kulkarni. 2020. MRAT: The Mixed Reality Analytics Toolkit. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376330

[48] Alex Olwal and Steven Feiner. 2004. Unit: Modular Development of Distributed Interaction Techniques for Highly Interactive User Interfaces. In *Proceedings of the 2Nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (Singapore) *(GRAPHITE '04)*. ACM, New York, NY, USA, 131–138. https://doi.org/10.1145/988834.988857

[49] Stephen Oney, Brad Myers, and Joel Brandt. 2014. InterState: A Language and Environment for Expressing Interface Behavior. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. ACM, New York, NY, USA, 263–272. https://doi.org/10.1145/2642918.2647358

[50] Etienne Peillard, Thomas Thebaud, Jean-Marie Normand, Ferran Argelaguet, Guillaume Moreau, and Anatole Lécuyer. 2019. Virtual Objects Look Farther on the Sides: The Anisotropy of Distance Perception in Virtual Reality. In *VR 2019 - 26th IEEE Conference on Virtual Reality and 3D User Interfaces*. IEEE, Osaka, Japan, 1–10. https://hal.archives-ouvertes.fr/hal-02084069

[51] Emmanuel Pietriga, Caroline Appert, and Michel Beaudouin-Lafon. 2007. Pointing and beyond: an operationalization and preliminary evaluation of multi-scale searching. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1215–1224.

[52] Ivan Poupyrev, Suzanne Weghorst, Mark Billinghurst, and Tadao Ichikawa. 1997. A Framework and Testbed for Studying Manipulation Techniques for Immersive VR. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (Lausanne, Switzerland) *(VRST '97)*. ACM, New York, NY, USA, 21–28. https://doi.org/10.1145/261135.261141

[53] Arnaud Prouzeau, Maxime Cordeil, Clément Robin, Barrett Ens, Bruce H. Thomas, and Tim Dwyer. 2019. Scaptics and Highlight-Planes: Immersive Interaction Techniques for Finding Occluded Features in 3D Scatterplots. In *Conference on Human Factors in Computing Systems (CHI 2019, May 4–9, 2019, Glasgow, Scotland Uk)*. Glasgow, United Kingdom. https://doi.org/10.1145/3290605.3300555

[54] Donghao Ren, Bongshin Lee, and Tobias Höllerer. 2017. Stardust: Accessible and Transparent GPU Support for Information Visualization Rendering. *Computer Graphics Forum* 36, 3 (2017), 179–188. https://doi.org/10.1111/cgf.13178

[55] Maxime Reynal, Euan Freeman, and Stephen Brewster. 2020. Avoiding Collisions When Interacting with Levitating Particle Displays. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/3334480.3382965

[56] Arvind Satyanarayan and Jeffrey Heer. [n.d.]. Lyra: An Interactive Visualization Design Environment. *Computer Graphics Forum* 33, 3 ([n. d.]), 351–360. https://doi.org/10.1111/cgf.12391 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12391

[57] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan 2017), 341–350. https://doi.org/10.1109/TVCG.2016.2599030

[58] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. 2016. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (Jan 2016), 659–668. https://doi.org/10.1109/TVCG.2015.2467091

[59] R. Sicat, J. Li, J. Choi, M. Cordeil, W. Jeong, B. Bach, and H. Pfister. 2019. DXR: A Toolkit for Building Immersive Data Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan 2019), 715–725. https://doi.org/10.1109/TVCG.2018.2865152

[60] Mark Simpson, Jiayan Zhao, and Alexander Klippel. 2017. Take a walk: Evaluating movement types for data visualization in immersive virtual reality. In *Proceedings of Workshop on Immersive Analytics (IA)*.

[61] Terece L. Turton, Anne S. Berres, David H. Rogers, and James Ahrens. 2017. ETK: An Evaluation Toolkit for Visualization User Studies. In *Proceedings of the Eurographics/IEEE VGTC Conference on Visualization: Short Papers* (Barcelona, Spain) *(EuroVis '17)*. Eurographics Association, Goslar Germany, Germany, 43–47. https://doi.org/10.2312/eurovisshort.20171131

[62] Teoman Ulusoy, Kurtis Thorvald Danyluk, and Wesley J Willett. 2018. *Beyond the Physical: Examining Scale and Annotation in Virtual Reality Visualizations*. Technical Report. Department of Computer Science, University of Calgary.

[63] J. A. Wagner Filho, C.M.D.S. Freitas, and L. Nedel. 2018. VirtualDesk: A Comfortable and Efficient Immersive Information Visualization Approach. *Computer Graphics Forum* 37, 3 (2018), 415–426. https://doi.org/10.1111/cgf.13430

[64] J. A. Wagner Filho, M. F. Rey, C. M. D. S. Freitas, and L. Nedel. 2018. Immersive Visualization of Abstract Information: An Evaluation on Dimensionally-Reduced Data Scatterplots. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 483–490. https://doi.org/10.1109/VR.2018.8447558

[65] Y. Yang, T. Dwyer, B. Jenny, K. Marriott, M. Cordeil, and H. Chen. 2019. Origin-Destination Flow Maps in Immersive Environments. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (Jan 2019), 693–703. https://doi.org/10.1109/TVCG.2018.2865192

[66] Yalong Yang, Tim Dwyer, Kimbal Marriott, Bernhard Jenny, and Sarah Goodwin. 2020. Tilt Map: Interactive Transitions Between Choropleth Map, Prism Map and Bar Chart in Immersive Environments. *IEEE Transactions on Visualization and Computer Graphics* (2020), 1–1. https://doi.org/10.1109/tvcg.2020.3004137

[67] Yalong Yang, Bernhard Jenny, Tim Dwyer, Kim Marriott, Haohui Chen, and Maxime Cordeil. [n.d.]. Maps and Globes in Virtual Reality. *Computer Graphics Forum* 37, 3 ([n. d.]), 427–438. https://doi.org/10.1111/cgf.13431 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13431

[68] Ji Soo Yi, Youn ah Kang, and John Stasko. 2007. Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics* 13, 6 (2007), 1224–1231.

Received July 2020; revised August 2020; accepted September 2020