The State-of-Art Platforms and Tools for Immersive Analytical Applications Development

Walid Chtioui*
University of Passau

Achraf Hebheb†
University of Passau

ABSTRACT

This paper presents an overview of the latest state-of-the-art platforms and toolkits which can be used to develop immersive analytics (IA) experiences. Initially, a short overview of Unity3D and Unreal Engine with a comparison of supported eXtended Reality (XR) features and third party tools is provided. After that, we present an overview associated with a critique of what we believe are the major XR development toolkits. For each toolkit we discuss the set of what we think are important features they provide or lack such as the support for collaboration, interactions, real-time data and large datasets. We then present a comparison study between these toolkits with a particular focus on performance, supported devices, accessibility and ease-of-use. A set of state-of-the-art tools and techniques, which can be used to extend aforementioned toolkits, covering collaboration, interaction and navigation topics are then provided. We have come to the conclusion that, although IA toolkits have come out of their infancy, we are still yet to see an IA toolkit that provides a unified workflow allowing for versatile visualizations, collaboration support, acceptable performance scalability, real-time data support and support for a plethora of XR and non-XR devices which has the potential to replicate the wide success of conventional visualization frameworks such as D3.js.

1 INTRODUCTION

2 PLATFORMS

Platforms refer to general-purpose frameworks that provide a uniform workflow allowing developers to create XR applications for a plethora of devices. Since IA experiences and AR/VR gaming applications share a lot of similarities, game engines provide excellent support for developing IA experiences. Two game engines are considered; Unity and Unreal Engine. A particular focus is given to Unity due to its wide adoptability for XR applications.

2.1 Unity

Unity has a wide adoption in the world of XR thanks to its unified workflow and support for various XR platforms. Unity supports an extensive set of XR vendor-specific software development kits (hereafter SDK) including: Apple's ARKit, Google's ARCore, Microsoft's HoloLens and OpenXR. Following the announcement of Apple's mixed reality (MR) headset Vision Pro in Apple's Worldwide Developers Conference (WWDC) 2023, Unity was announced to provide native support for Vision's Pro operating system VisionOS [1]. Unity also provides a set of XR packages that are built on top of these vendor plugins to add application-level development tools [11]. For instance, AR Foundation is an industry-standard framework that provides support for various AR features such as: object tracking and plane detection. Unity also provides XR Interaction Toolkit package which is a high-level, component-based interaction system. The package also includes XR Device Simulator which is a simulator that allows user input from conventional input devices (a keyboard, a mouse or a controller) to drive XR headset and controllers in the Unity scene view. This may be useful for debugging on a wide range of XR devices without having to actually try them. Unity also benefits from open-source external XR packages such as the Mixed Reality Toolkit for Unity (MRTK) [1] which is designed to further accelerate cross-platform MR development in Unity.

2.2 Unreal Engine

Unreal Engine is an industry-standard software for creating hyper-photorealistic 3D/2D experiences. Thanks to its unified workflow across numerous XR vendors such as Oculus, Microsoft, Vive, and PlayStation, it provides the required assets to develop XR experiences This engine, like Unity, comes with the core comprehensive collection of XR SDKs such as Google's ARCore, Apple's ARkit, Magic Leap Microsoft's HoloLens, OpenXR, and Oculus. However, Unreal Engine comes with a less-than-ideal XR documentation relative to Unity. This documentation does not provide comprehensive manuals for certain XR features. As a result, we cannot guarantee that these functionalities are supported. Moreover, Unreal Engine lacks official support for third-party XR plugins as is the case with Vuforia.

2.3 Comparison

Figure 1 provides a comparison between the previously discussed platforms in terms of support for vendor-specific SDKs and a set of features.

3 TOOLKITS AND FRAMEWORKS

The term toolkit refers to development environments that are tailor-made for IA experience development purposes. They provide, among other things, high-level tools for authoring visualizations, built-in interactions and support for multiple major XR devices.

3.1 DXR Toolkit

Sicat et al. proposed DXR [10]; an IA toolkit built on top of Unity that allows fast prototyping and iteration for non-experienced users; i.e. users with no or little programming knowledge in XR and Unity. Alongside the data input, DXR takes a specification file written in JavaScript Object Notation (hereafter JSON) from which visualisations are created. The specification file is described in Vega-Lite declarative grammar [8] (only what should be achieved has to be provided, not how) making it suitable for users with no programming experience to rapidly realise immersive visualisations. This configuration file can be edited in a separate text editor or through the use of a GUI with pre-configured set of parameters. DXR also provides built-in specification templates for common visualisations such as: Scatter plots and bar charts. This further extends the scope of users to include those without any technical experience. Although the authors claim that DXR provides suitable flexibility, the scope of that flexibility seems to be limited to providing, among other things, custom graphical markers, custom visualization channels - i.e. a visualisation parameter affected by a data dimension - and other visualisation-type specific properties. That limits users to a

*e-mail: walid.chtioui@ensi-uma.tn
†e-mail: habhabachref@gmail.com

[1] https://github.com/MixedRealityToolkit/MixedRealityToolkit-Unity

	Platform	
SDKs	Unity 2022 LTS	Unreal Engine 5.3
ARCore	X	X
ARKit	X	X
Magic Leap	X	X
Microsof HoloLens	X	X
OpenXR	X	X
Oculus	X	X
WebXR	X	
VisionOS	X	
AR-Specific Features		
Plane Detection	X	X
Object Occlusion	X	X
Environment Probes	X	X
Face Tracking	X *(Android& iOS only)	X *(Android & iOS only)
Image Tracking	X	
Object Tracking	X *(iOS only)	?
Body Tracking	X *(iOS only)	?
Camera Intrinsics	X	X
Meshing	X	?
XR Features		
High-level XR Interactions	X	?
XR Input Simulation	X	?
Vuforia Support	X	

Table 1: Per-platform supported SDKs, AR features and 3rd party tools. *?: feature could not be found in engine's official documentation.

templated and common set of visualisations such as scatter plots, bar charts and radial bars. There is also no mention of real-time data support thus limiting the use case of DXR to offline data only.

As the authors have explicitly mentioned, DXR is meant for prototyping and exploring designs, it is not designed to handle visualisations of large datasets. On HoloLens, for datasets with more than approximately a thousand item, suboptimal - less that 60 frames per second (hereafter FPS) - performance has been observed. Nonetheless, the authors argue that DXR can still be useful for quickly and cheaply prototyping large dataset designs before moving to specialized, optimized and detailed implementations.

3.2 IATK Toolkit

Maxime et al. introduced IATK [3]; an open-source [2] software package for Unity that provides both a high-level Unity-editor-integrated GUI for simple authoring and a low-level C-sharp and JavaScript API for fine-grained authoring and extending the visualizations.

To some degree of similarity to DXR, IATK relies on a high-level declarative grammar of graphics by providing a composable grammar of visualization primitives alongside a high-level interface for rapid prototyping and iterations. What sets IATK apart, is that it was designed with scalability in mind, a focus on large and complex multidimensional datasets and a focus on user interactions. The toolkit's authors claim that it can render millions of items thanks to its use of efficient GPU shader code. Also, contrary to DXR, IATK does not support declarative configurations, instead it relies on a Unity editor GUI or C-sharp API code that make use of a composable grammar to author visualizations.

Unlike DXR and other toolkits built on top of Unity, IATK does not render the datapoints (here we do not mean point as in a geometrical point but as a representation of a data entry) as Unity game objects and use expensive-to-update-at-large-scale object attributes as visualization channels. Instead, all the datapoints are visualized within one game object where each data point is encoded into a unique vertex by mapping data attributes, such as position, color and size, into vertex components such as vertex UV coordinates, vertex normal vector and vertex color. This way, actual datapoint geometries are created on the GPU resulting in a potentially more efficient rendering process. This however greatly limits the customizability of datapoint marks and the choice of visualization channels especially for novice users.

According to the authors' performance statistics, on VR less than 90 FPS is observed at two million datapoints and on AR less than 60 FPS is observed at just a thousand datapoints. However, one can subjectively claim that the performance on AR HoloLens headset remains acceptable up-to ten thousand datapoints at which 41 FPS is achieved. Although the authors provided FPS statistics for Oculus CV1, Meta2 and HoloLens devices, no performance statistics were provided for the alternative game-object-based datapoint approach to provide a performance reference point.

IATK integrates an interactive visualization model within its visualization components that allows a set of interactions including filtering, brushing and linking (1 (8)), details on demand, animated transitions and attribute-based animations. These interactions are implemented in the vertex Shader part of the rendering pipeline which leverages the high parallelism nature of the GPU(s) making them particularly responsive and efficient at handling large datasets.

IATK's high-level GUI provides just a small set of built-in visualization types such as a 3D\2D Scatterplot (1 (3)), a parallel coordinates plot (1 (5)) or a 3D\2D Scatterplot matrix (1 (4, 6, 7)). It also only provides a small set of datapoint geometries. To extend these, one has to use the provided low-level API which might limit expressiveness for non-experienced users.

The authors did not mention any support for neither real-time data visualization nor local or remote collaboration. It is also worth men-

[2] https://github.com/MaximeCordeil/IATK

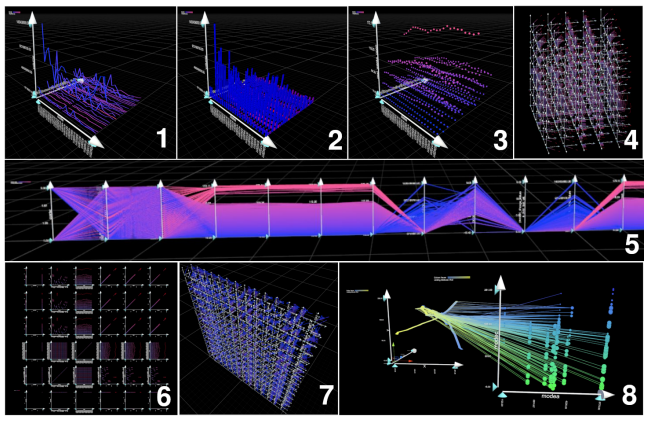

Figure 1: IATK supported visualization types. (1) 3D connected dots. (2) 3D bar chart. (3) 3D Scatterplot. (4, 7) 3D Scatterplot matrix. (5) Parallel Coordinates Plots. (6) 2D Scatterplot matrix. (8) Linked visualizations. The images are from [1] and are in the public domain.

tioning that for VR, only Scatterplot visualization type is supported. This greatly limits the expressiveness for VR users.

3.3 VRIA Toolkit

Peter et al. introduce VRIA [2]; a free and open-source [3] framework for building IA experiences in VR. Unlike the aforementioned toolkits, VRIA is not built as an add-on on top of a game engine. Instead, it is built upon open-standard Web-based frameworks such as WebVR, A-Frame, React and D3.js. All of which are mature, open-source and widely used Web-based frameworks and libraries. This allows VRIA applications to be accessible by a plethora of VR and non-VR devices.

The toolkit is designed to be accessible to novices and experts alike. Similar to DXR, VRIA makes use of a declarative grammar similar to Vega-Lite [8] which allows users to create custom visualizations based on a configuration file. However, this visualization configuration file is only required for basic functionalities. For extra custom functionalities such as a custom set of visualization channels, interactions, graphical marks or visualization types, more experienced users are advised to use the provided low-level API. VRIA provides The VRIA Builder; a Web application intended for beginners that integrates a GUI and a 3D scene view to rapidly prototype visualization designs with instant feedback without having to leave the browser. It is worth mentioning that, unlike DXR, this customization GUI is not part of the VR scene and cannot be viewed withing the VR headset. Only the scene view can be experienced immersively. However, the authors mentioned their willingness to add *in-situ* GUI so that users can build and prototype visualizations iteratively without having to remove the headset and switch back to desktop screen.

Thanks to its composable structure, the toolkit can be integrated into other existing 2D or immersive visualization applications. For instance, VRIA's visualizations can be overlaid on top of another A-Frame scene. VRIA has support for collaborative immersive analytics through either the high-level networking abstraction layer provided by the open-source Networked A-Frame component or by using the provided API with lower-level networking libraries.

In terms of data input, the toolkit only supports tabular data in the form of JSON or CSV thus real-time data visualization is not supported. The authors expressed their willingness to add support for other forms of data models such as geospatial GeoJSON, network and relational models.

[3] https://github.com/vriajs

The authors stated that VRIA offers the option to integrate D3.js visualization within its IA experiences. D3.js is web-base JavaScript visualization library that provides a low-level approach to author graphics and powerful mechanisms to transform and manipulate data. Since D3.js is widely adopted, integrating it within VRIA allows for easier usability and quicker authoring of 2D visualizations. However, there was not a description of which D3.js functionalities and visualizations are integrable nor any guidance on how to integrate it within the VRIA framework.

More experienced users seeking for a more custom experience are forced to deal with two separate tools to author custom visualizations; the visualization configuration file specified in a declarative language and the API specified in JavaScript. We believe that it would be easier for the end users if VRIA provided more customizability through the visualization configuration file instead of just providing very basic functionalities. Although it is understandable that A-Frame simplifies the process of creating VR experiences on the web relative to the lower level Three.js library, we question its necessity. A-Frames' functionalities could be directly implemented in Three.js thus reducing and simplifying the VRIA stack. The authors stated their intention to work at the Three.js level directly without the usage of any further abstraction libraries.

Whenever 3D graphics are mentioned in the context of web browsers, performance implementations are one of the most worrying aspects. This is especially true for Web-based VR applications where two images have to be rendered each frame preferably at 90 FPS to avoid motion sickness. VRIA's authors provided visualization benchmarks on a desktop monitor, a desktop Oculus Rift CV1 and a smartphone. A scatter-plot visualization type was used with sphere graphical marks. The exact set of visualization channels used was not mentioned neither were attributes of the input data. On the desktop monitor and Oculus Lift HMD, performance drops significantly after one thousand data points. On smartphone performance starts dropping at one hundred data points. This proves that VRIA is not suitable for large dataset visualizations. The authors claim that the performance observed for the HMD is similar to that of IATK for the HoloLens device. But the HoloLens uses its own, much less powerful, hardware while VRIA performance benchmark used the Oculus Lift CV1 VR HMD which relies on a separate desktop for expensive rendering operations. We therefore question such comparisons of performance benchmarks. This is later Web-based XR solutions do not compete, in terms of performance, with game-engine based solutions.

VRIA only supports Cartesian plots but the authors expressed their willingness to add support for other coordinate systems including geographical and spherical coordinates. Moreover, although the authors showcased the ease of integrating VRIA with AR.js to target AR experiences, the absence of an out-of-box support for VRIA still limits AR accessibility only to experienced users.

3.4 RagRug Toolkit

Philipp et al. presented RagRug [5]; An open-source [4] toolkit built on top of IATK for *situated analytics*, i.e., real-time data-fed interactive immersive visualizations connected to a meaningful physical object in close proximity to the user (hereafter we refer to such an object as a *referent*). For instance, a facility manager equipped with an AR HMD, when in proximity to an air conditioner with multiple Internet-of-Things (IoT) sensor devices monitoring temperature and other parameters, multiple real-time visualizations pop up (2). Since it only targets situated-analytics usecases, RagRug is an AR-only toolkit. A major requirement for IA toolkits is to provide support for a wide variety of input and output modalities spanning VR/AR and potentially conventional 2D screen displays. RagRug aims, on top of that, to achieve a broad coverage of IoT

[4] https://github.com/philfleck/ragrug

Figure 2: Example of situated analytics using RagRug toolkit. Multiple visualizations are shown and are fed real-time Images from [4] and is in the public domain.

temperature data from nearby IOT sensor devices.

devices. It also intends to be used for rapid prototyping by providing runtime interpretation of application logic.

The core design goals and features of RagRug include:

- **Providing 3D Visual Encoding Capabilities** - RagRug provides 3D *visual encoding* capabilities, i.e., ways of mapping data dimensions into 3D visual structures. These capabilities are inherited from IATK and the visual representations are the same.
- **Making Visualizations Context Aware** - RagRug provides visualizations that are context-aware with regards to changes in the real world. This reduces the need to explicitly specify every minute detail while the user is emerged in a physical task. For instance, If a referent is moved, the attached visualization will move along with it or if the lighting conditions change then visualization color will adapt to such new condition or if the user performs an invalid action, a notification is adequately shown. RagRug makes use of IoT devices for collecting data about the environment and a cloud server for, among other things, the IoT application logic and depositing data in a database backend so that the client can dynamically query it.
- **Supporting a Comprehensive Physical-Virtual Model** - Information about the characteristics of referents such as location, shape or identity are not hard-coded and are instead queried from a suitable database. That way the client obtains relevant data dynamically. This is important for correctly linking visualizations to referents.
- **Making the Visualization Pipeline Reactive** - The pipeline provided by RagRug is not re-evaluated after every external change. For instance, when new data comes, visualizations will be updated. Or when a device is turned off, marks in a visualization will change to reflect that. IATK lacks support for either real-time data (since it expects a static dataset input) or for receiving events from external sources. To address this, RagRug provides an extension on top of IATK to reactively update data points on network events locally and externally.
- **Supporting Situated Authoring of Visualizations** - Since testing situated analytics requires physical interactions with referents, not allowing for situated authoring may result in a cumbersome development experience. The authors propose runtime interpretation of application logic which allows designers to modify every aspect of the system without having to shut it down. To address the lack of in-situ authoring from IATK, The authors make use of PowerUI [5], an open-source JavaScript extension for Unity that allows for runtime UI authoring using CSS, HTML and JavaScript.

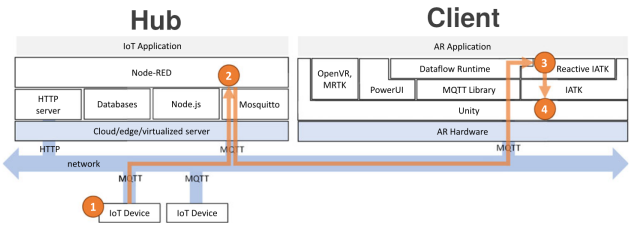

Figure 3: RagRug stack. (1) Data is acquired from IoT sensors using MQTT. (2) Data filtering. (3) Real-time data is sent to IATK as a response to a query. (4) Visualizations are rendered/updated in Unity.

RagRug consists of two standalone platforms, one for experiencing immersive analytics visualizations and interacting with them (referred to as *client*) and one for IoT applications and databases (referred to as *hub*). As illustrated in (3) the two platforms make use of the widely adopted and lightweight MQTT [6] protocol to communicate data and events. Another core design of the toolkit is its heavy reliance on a uniform dataflow programming model. To achieve that, the hub makes use of Node-Red [7]; a low-code programming tool for event-driven applications. The client also relies on the dataflow graphs of Node-Red. These are provided to the client, alongside other JavaScript runtime dependencies, at initialization time. The dataflow model expands the range of potential RagRug users to include those with low programming experience. RagRug emphasises on making JavaScript its unified programming language by making use of extensions that provide a JavaScript wrapper interface that calls the native interfaces under the hood (e.g., PowerUI[8] which allows for runtime Unity UI authoring using JavaScript instead of the native Csharp programming interface). Since the toolkit is event-driven, it supports collaborative situated analytics.

Although we do understand the importance of having a unified programming model, We find the use of PowerUI, for the sole reason of providing a JavaScript-based tool for authoring Unity runtime UI, questionable. Unity provides a variety of built-in runtime UI authoring tools that are mature, better documented, and proven by the community. Unfortunately, these tools rely on Csharp for their logic which if adopted, the unified programming model is no longer satisfied. Another concern is that PowerUI has been deprecated from the Unity Asset store and the open-source GitHub repository has not been active, except for a README file change, for more than seven years [9]. We therefore argue for relaxing the constraint of a unified

[5] https://github.com/Kulestar/powerui
[6] https://mqtt.org/
[7] https://nodered.org/
[8] https://powerui.kulestar.com/
[9] https://github.com/Kulestar/powerui

programming model in favor of less technology-stack complexity and reliance on established built-in tools.

We also believe that the reliance on low-code, visual-programming tools such as Node-Red can hinder the scalability of RagRug. This would have not been an issue if RagRug provided a low-level API for users looking for a more advanced, out-of-the-usual usecase.

In-depth statistics for the performance of the RagRug toolkit was not provided. Instead, a single statistic that used a HoloLens 2 as a client, a notebook computer as a hub and a 5GHz-band wireless network was provided. However, the event propagation time was the only provided metric by this statistic. This leaves the question of the usability of the toolkit in production environments with various data loads largely unanswered.

The authors did not provide any statistics about user feedbacks other than that they are mainly researches. We believe that gathering feedback from a wide range of users, both client-side users and developers, is important in determining the validity of the claimed characteristics of the toolkit (e.g., ease of usability, ease of customizability, etc.).

3.5 Comparison

4 Prototypes

While tools refer to systems that aim to allow developers to create IA experiences, prototypes are applications that target end-users by providing IA experiences attempting to solve particular IA-related and/or domain-specific problems. Through the feedback of end-users, prototypes may provide interesting insights on how IA tools should address particular challenges.

4.1 Fiesta

Benjamin et al. built FIESTA [7]; an open-source [10] team-based analysis prototype that allows users to create, manipulate and share data visualizations in an unconstrained and physically co-located VR-only environment. The authors state that the main objective behind developing FIESTA is to study how groups collaboratively explore data in a free-roaming VR environment.

FIESTA is built on top of IATK. The authors justify the usage of IATK for performance reasons since multiple users can each produce multiple visualizations. A $16m^2$ (4m x 4m) room is used to conduct the study. Each user is equipped with a VR headset and a backpack containing a laptop to which the headset is connected (4 (1)). To track the headsets' positions, four trackers are positioned in the room; one in each corner. An additional desktop PC manages the users' laptops using the Unity networking engine Photon. Although the study focuses on physically co-located IA environment, the authors state that the research prototype also supports remote collaboration.

The core design goals and features of FIESTA are:

- **Baseline Visualizations** - The prototype provides three types of 2D and 3D visualizations: scatterplots, faceted scatterplots (4 (6)) and line charts.
- **Baseline Interactions** - FIESTA relies on standard interfacing techniques across desktop and VR. It provides grasping techniques for interacting with UI elements at close range and ranged pointing techniques using a laser pointer for long range (4 (7)).
- **Baseline Authoring** - Each user is provided with a movable panel to author visualizations (4 (2, 3, 4, 5, 6)). The panel is split into two sections: on the left, a GUI is provided to map data dimensions to visualization channels, choosing the faceting dimension and adjusting the number of facets; the right panel is used to show produced visualizations.

[10] https://github.com/benjaminchlee/FIESTA

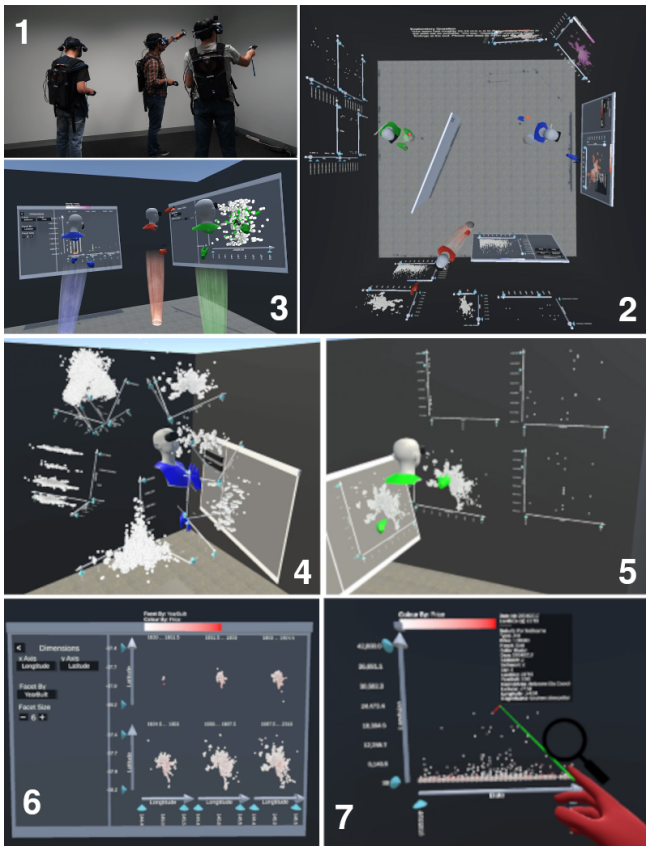

Figure 4: (1) Real view of participants using the FIESTA system. (2) Top-down view of a virtual scene in which three participants are conducting visual analytics tasks. (3) A virtual view from the perspective of an observer. (4) Egocentric layout. (5) Planar layout. (6) Authoring panel with a faceted scatterplot on its right interface. (7) Details on demand to easily inspect data records. All images are taken from [7].

- **Tearing out Visualizations** - Visualizations can be cloned from the panel by grasping and pulling them away. These cloned visualizations can then be placed independently in the 3D space (4 (2, 4, 5)). Resizing and rescaling operations are supported through grasping interactions on widgets placed along visualization object's axis. Visualizations can also be brought back into the panel for further editing or be destroyed through a throwing-towards-ground interaction.
- **Linked Brushing** - Private and shared brushes are provided for each user. The private brush allows for selections that are only visible to the user while the shared brush allows for selections which are visible to all users. Brushing is linked across multiple visualizations through color channel, i.e. contrary to other tools where lines are used to visualize linked datapoints.
- **Pointer and Details on Demand** - A pointer is provided and is activated through controller's trigger. Details-on-demand tool of the nearest datapoint is enabled through a touchpad input (4 (7)). The pointer can also be used to interact with the panel's interface through point-and-click interaction for buttons or point-and-drag interaction for sliders and pickers.
- **Free Roaming Shared Environment** - To emulate a physically co-located collaborative environment, Each user is represented by a uniquely colored avatar and a floating nameplate (4 (3)). The same unique color of the user's avatar is used

for coloring shared brush selection. There is no ownership system in FIESTA; everything is seen and interactive by all users except the private brush tool.

- **Room and Surface Affordances** - To explore how users naturally use surface-based functionalities in collaborative IA environments to solve visual analytics tasks, a four-wall room with a square tabletop in the center is used. The virtual walls act as a boundary to prevent users from hitting the actual room walls. Releasing visualizations close to the virtual surface of the table top make them rest on top of it.

A total of 30 participants were recruited for the study with all but one having a background in computer science. The participants managed to perform visual analytics tasks and remained engaged with these tasks for 45 minutes on average. This proves that collaborative visual analysis is possible in VR. Participants made use of 3D visualizations although their use was not emphasized. They came up with interesting ways to view them such as: scaling a 3D visualization to room scale and standing in the middle of it; laying 3D visualizations around them in an egocentric layout (4 (4)) and looking at them from an axis-aligned viewpoint. With 2D visualizations, participants preferred placing them in a grid-like layout on the walls. This made them more presentable to others at all times contrary to 3D visualizations where users struggled to take the perspective of observers into consideration when viewing them. Therefore, IA tools should explore methods to facilitate the presentation of 3D visualizations to other users. Another interesting observation is that 3D visualizations heavily influenced the way 2D visualizations are organized. For instance, in the presence of 3D visualizations, these 2D visualizations were placed in an egocentric layout. These observations demonstrate the importance of a consistent UI design that supports both 2D and 3D visualizations. The study findings may also justify the need to provide different placement tools for 2D and 3D visualizations. Furthermore, although the usage of surfaces is completely optional in FIESTA, participants made use of the walls to neatly place their 2D visualizations. However, they saw no benefit in using the central table top for visualization placements. This demonstrates that users are influenced by environment configuration only if there is a tangible benefit to its usage. The majority of users mixed between individually and collaboratively working although both modes were optional in FIESTA. This proves that providing support for both working modes is invaluable. The study also showed that participants did not interact with objects that, except in cases where such interactions were expected, did not belong to them. However, the reason behind that could be avoiding physical collisions with other users. Therefore a study with remotely co-located environment is needed to further study these behaviours.

Although the usage of surfaces is optional in FIESTA, they were inherently predefined - the users did not have the option to arrange such surfaces. A similar system in which they have to ability to define their own surfaces may further expand these findings. Furthermore, FIESTA is a VR-only prototype therefore these findings may not be applicable to IA experiences where AR is used.

4.2 Uplift

Barrett et al. proposed Uplift [4]; an in-place collaborative visual analytics prototype targeting users with diverse expertise in the domain of microgrids. Uplift is designed for casual visual analysis use-cases; i.e. to be used to easily identify, in a relatively short time, key patterns in complex visualized data. The requirements for the prototype were initially provided and subsequently modified, through multiple feedback sessions, by a wide range of stakeholders including microgrid project and energy systems experts.

The main design goals of Uplif that were identified through multiple stakeholders feedback sessions are:

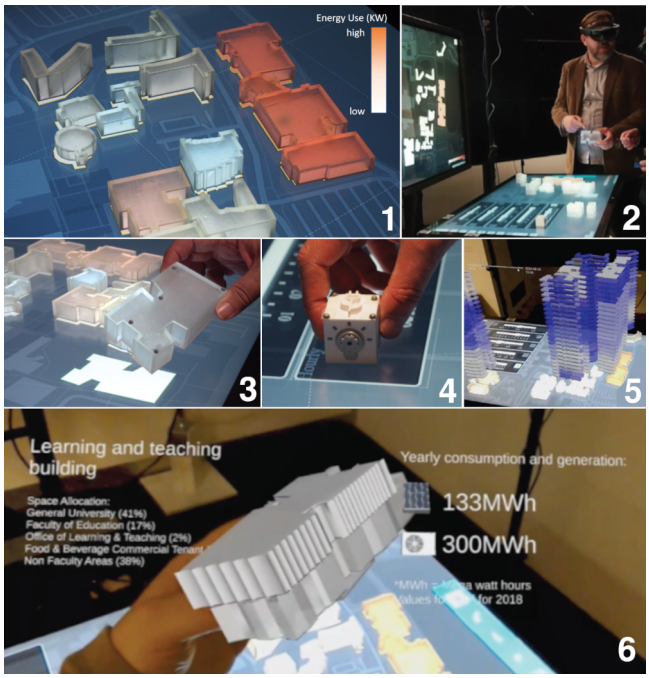

Figure 5: A showcase of Uplift prototype. (1) Translucent scaled-down models of campus buildings placed on top of the colored tabletop to reflect energy consumption. (2) Users gathering around the central tabletop supported with a large 2D display. (3) A user picking up a scaled-down building model of the campus. (4) A user interacting with a tangible widget affecting a slider which in turn affects time granularity and therefore the number of slices in (5). (5) AR for visualizing data above and around the central tabletop, in particular hourly energy consumption is visualized as 2D slices laid over buildings. (6) A picked-up campus scaled-down building model overlaid with relevant data by AR. Images from [4].

- **Walk Up and Use**: The system should be inviting, intriguing and compelling.
- **Low Technical Barrier to Entry**: Easily usable by users with a wide degree of technical knowledge.
- **Supports Multiple Participants**: Provide flexible collaboration support to small and large groups alike.
- **Supports Analytical Sprints**: Provide support for short analytical activities.

Uplift relies more than just AR headsets to bring casual collaborative visual analytics to a multitude of microgrid stakeholders. A tabletop display showing a geographical map of the campus grid is used as a central platform where users are supposed to gather around and interact with widgets placed on top of it (5 (2)). Uplift also makes use of tangible widgets which are physical and interactive elements that control visualization parameters (for instance by affecting sliders) (5 (4)). The prototype also relies on scaled-down physical models of buildings that are translucent which allows the color of the surface on which they are placed to be used as an appealing visualization channel (5 (1)). If these models are picked up, AR will overlay relevant data about them (5 (6)). AR is used to display multiple 3D data types on top of the tabletop and 2D graphs alongside legends around it (5 (5)). On top of these, Uplift uses a large display to either replicate the content of the tabletop or show additional visualizations (5 (2)).

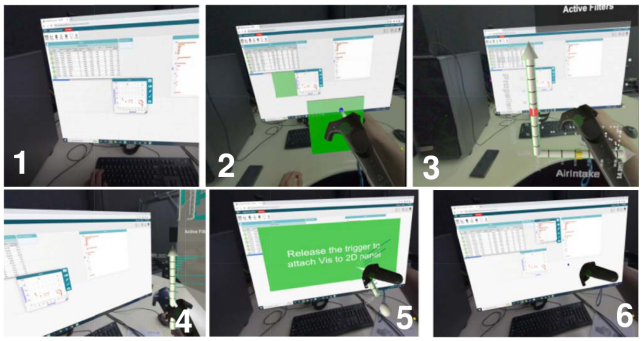

Figure 6: HybredAxes context switch processes. (1, 2, 3) Pulling data/visualization from a 2D screen. (4, 5, 6) Pushing a visualization to a 2D screen from an immersive AR environment.

Through the feedback of 16 participants who tried the prototype, Uplift was proven to be potentially useful for microgrid-related data analytics.

Although Uplift was designed for microgrid-related systems, the authors claim that its applicability domain can be extended to include other fields that rely on analysis of complex spatial data for sense-making such as the construction industry. However, the use of a wide range of technologies and gadgets makes Uplift a specialized solution that we believe is not yet ready for wide deployment. For instance, on top of using Vuforia for tabletop tracking, Uplift uses the motion capture Vicon system [11] with a proprietary tracking software and four cameras to track the tangible widgets and the buildings. However, since no proof was provided for the usefulness of overlaying information when a building model is picked, we question the usage of object tracking for this purpose. This is particularly true if such feature requires the addition of expensive gadgets and proprietary software. For instance, model position is already embedded in the tabletop map visualization and that can be used by AR to overlay visualizations above without the need of object tracking. We also argue that the use of tangible widgets to control visualization parameters does not justify the usage of additional gadgets and software required to track them.

Uplift expects a static data input therefore, although the topic of real-time monitoring of the microgrid was seen as beneficial for operators by expert stakeholders, the tool does not support real-time immersive analytics.

5 Collaboration Tools and Techniques

6 Interaction Tools and Techniques

Interaction is an important component of information visualization. It has even been argued that it provides a way to overcome the limits of representation and augment a user's cognition [12].

7 Navigation Tools and Techniques

8 Transition and Animation Tools and Techniques

TODO: add introduction with suitable references for importance of animations and cross-virtuality transitions.

8.1 HybridAxes Tool

Mohammad et al. introduced HybridAxes [9]; an extension of IATK toolkit that allows for a smooth interoperability between 2D desktop visualizations and immersive reality experiences (6).

HybridAxes aims to provide a balanced feature set and performance between the two ends of the Reality-Virtuality continuum while giving the users the ultimate choice to choose which environment to work in. To avoid disruptions of the user's flow when context switching, the tool also aims to provide a smooth transition by reducing visualization generation delays. It claims to do that by anticipating the user's interactions in both modes and preemptively generating visualizations in the background. Moreover, the authors stated that providing a clear signaling of transition status either through visual (6 (2, 5)) or controller-related feedback is a core design requirement for the tool. Keeping the same values for the visualization channels, if possible, when transitioning a visualization between the two experiences is also stated as a core design goal. HybridAxes should also provide support for synchronized cross-virtuality brushing and linking. For instance, a user may have a tabular data on a 2D desktop alongside a 3D scatterplot in the immersive environment. When highlighting an entry in the table, associated datapoints in the scatterplot should also be highlighted. HybridAxes, aims to also favor desktop for interactions that require the detailed manipulation of text entries. The choice of interaction method, either through free-hand or controller-based interactions, is given to the user.

In terms of implementation, HybridAxes adopts CODAP; an open-source [12] visual analytics tool, alongside a plugin that enables the desktop system to receive commands via a Websocket interface. A Node.js-based Websocket server is used to relay the messages between Unity and the desktop system. We question the use of these extra tools and frameworks since we believe that transitioning between immersive and 2D Unity visualizations is superior to the suggested workflow. Our main argument is that this way all visualizations are established within one framework - Unity - which therefore allows for a more versatile, accessible and easier-to-use toolkit. The tool provides support for the same set of visualizations as IATK mainly: Scatterplots, Parrallel Coordinate Plots and Bar Charts.

In terms of performance, the authors claim that anticipating user actions has resulted in smoother experience. However, no comparative performance measurements were provided to back this claim.

In terms of interactions, three types are supported:

- **Pull visualizations from the 2D desktop** - by using free-hand gestures or the controller(s) (6 (1, 2, 3)).
- **Push visualizations from the immersive environment back to the 2D desktop** (6 (4, 5, 6)).
- **Brush datapoints on either side of the environments and see their counterparts in the other side get highlighted**.

The authors hypothesize that enabling visual analytics users to smoothly context switch between a 2D screen and 3D immersive experience could improve their sensemaking abilities. A limited study of the usefulness of the tool has been conducted on just four participants where they analyzed the data in three scenarios: 1) just using 2D desktop screen, 2) using a hybrid desktop/AR system and 3) just using the AR system. The authors claim that the study results align with their hypothesis. However, neither metrics nor sufficient details were provided to prove this claim. Nonetheless, the authors mentioned their willingness to perform a more rigorous study in the future.

9 Discussion

Throughout our survey, we have learned that current state-of-the-art IA toolkits have, to some degree, properly addressed a set of important challenges. These challenges include:

[11] https://www.vicon.com/

[12] https://codap.concord.org

- **Uniform Workflow** - Proposed tookits managed to provide uniform streamlined workflows that made it easier for users to make IA experiences.
- **Ease-of-use** - IA community provided tools for both: novice users with little or no XR and programming experience; expert users to allow them to expand the toolkit. The community also experimented a lot with high-level grammars.
- **Representation** - Community provided 2D and 3D visualizations for common plots such as: scatterplots, bar charts and parallel coordinates plots. In particular, visual representation of connected datapoints between plots was also focused on.
- **Interaction** - Community managed to provide interaction tools that handle, among other things, repositioning, resizing, selection, details-on-demand, brushing and filtering. These are essential for better immersive analysis experience.
- **Collaboration** - Community provided valuable insights about the behavior of participants in collaborative IA environment both in co-located space and remotely. The community also experimented with potential new features and documented their usefulness. These features include revisiting interaction history, user avatars and shared brushing tools.
- **Adequate Authoring Tools** - Adequate authoring tools for visualizations in the form of GUIs or/and configuration files were provided.
- **Low-level API for Extensions** - In most discussed toolkits, a low-level API allowing for the extension of functionalities is provided. This has allowed newer toolkits to build upon a customized version of a previous one [5].

Although the aforementioned toolkits and prototypes prove that the field of immersive analytics is past its infancy stage, there are still many open challenges that are not yet properly or not at all addressed. We believe that support for the following features by current state-of-the-art IA toolkits is still lacking:

- **Versatile Data Input Model** - Community should keep assumptions about the input data as minimal as possible. In particular, real-time data should be supported. Moreover, methods for efficiently updating visualizations should be adopted as well (e.g., when a new datapoint is received, when visual characteristics of some datapoints change, etc.).
- **In-depth Performance Statistics** - Community should focus more on addressing the core challenge of performance. Convincing performance statistics are still lacking.
- **Simple Technology Stack** - Community should focus more on simplifying the technology stack and avoiding characteristics that may hinder the wide adoption of the IA toolkit (e.g., usage of many tools from different vendors, usage of no-longer-maintained third party tools, usage of expensive or hard-to-get hardware, etc.).
- **Usage of Open-Standard Technologies** - Community should focus more on using open-standard framework/libraries. We do believe in particular that the community should experiment with the low-level XR library OpenXR instead of game-development frameworks.
- **Clear Licensing Terms** - For a successful wide adoption, licensing terms for toolkits should be clarified and complex licenses should be avoided.
- **Extensive Customization Support** - Community has, to some degree, successfully addressed the customizability aspect of IA toolkits. Nonetheless, we argue in favor of providing a low-level API that allows for a wider range of configuration.
- **Built-in Collaboration Support** - Community should work more towards adding built-in collaboration support (i.e., treat collaboration as a first class citizen). Moreover, aspects of collaboration should be customizable (e.g., user avatars, shared tools, etc.).
- **Portability to Conventional 2D Screens** - Community should work more towards adding support, on top of XR technologies, to conventional 2D screens. We believe this is important since XR devices are still not yet widely adopted.
- **Portability to the Web** - Since web-based conventional visualization libraries have seen a wide adoption, the community should work more on adding web support.
- **In-situ Authoring Tools** - Community should work more on adding in-place (i.e., in real use-case environment) authoring tools.

API with Bindings to High-level Programming Languages - The bindings should be good

Therefore we argue in favor of a visualization-tailored tool that is built on top of open-standard XR technologies. We do particularly firmly believe that an open-source and free-to-use - i.e. provides a permissive free software license - tool built on top of the open-standard OpenXR API can mimic the success of conventional 2D visualization libraries such as D3.js. Unfortunately, we have not come across such a framework and the closest to that in terms of reliance on open-standard technologies would be VRIA.

10 Conclusion

We presented an overview of the platforms, tools and prototypes in the last six years which can be used to develop IA experiences. Although IA tools have certainly come out of their infancy with several examples of tools addressing particular IA challenges well, we have come to the conclusion that there is not yet any IA toolkit that has the same level of versatility, visualization-tailored, ease-of-use and support of various data inputs and streams such as the widely successful 2D visualization libraries like D3.js. This is consistent with the latest IA survey from 2021 [6] where no tool meeting these criteria was mentioned.

References

[1] Apple. *Bring your Unity VR app to a fully immersive space*. `https://developer.apple.com/videos/play/wwdc2023/10093/`. Last accessed 03 December 2023. 2023.

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

[3] Maxime Cordeil et al. "IATK: An Immersive Analytics Toolkit". In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE. 2019, pp. 200–209.

[4] Barrett Ens et al. "Uplift: A Tangible and Immersive Tabletop System for Casual Collaborative Visual Analytics". In: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2021), pp. 1193–1203. DOI: `10.1109/TVCG.2020.3030334`.

[5] Philipp Fleck et al. "RagRug: A Toolkit for Situated Analytics". In: *IEEE Transactions on Visualization and Computer Graphics* 29.7 (2023), pp. 3281–3297. DOI: `10.1109/TVCG.2022.3157058`.

[6] Adrien Fonnet and Yannick Prié. “Survey of Immersive Analytics”. In: *IEEE Transactions on Visualization and Computer Graphics* 27.3 (2021), pp. 2101–2122. DOI: 10.1109/TVCG.2019.2929033.

[7] Benjamin Lee et al. “Shared Surfaces and Spaces: Collaborative Data Visualisation in a Co-located Immersive Environment”. In: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2021), pp. 1171–1181. DOI: 10.1109/TVCG.2020.3030450.

[8] Arvind Satyanarayan et al. “Vega-Lite: A Grammar of Interactive Graphics”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 341–350. DOI: 10.1109/TVCG.2016.2599030.

[9] Mohammad Rajabi Seraji and Wolfgang Stuerzlinger. “HybridAxes: An Immersive Analytics Tool With Interoperability Between 2D and Immersive Reality Modes”. In: *2022 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*. 2022, pp. 155–160. DOI: 10.1109/ISMAR-Adjunct57072.2022.00036.

[10] Ronell Sicat et al. “DXR: A Toolkit for Building Immersive Data Visualizations”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 715–725. DOI: 10.1109/TVCG.2018.2865152.

[11] Unity3D. *XR packages*. https://docs.unity3d.com/Manual/xr-support-packages.html. Last accessed 03 December 2023. 2023.

[12] Ji Soo Yi et al. “Toward a Deeper Understanding of the Role of Interaction in Information Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1224–1231. DOI: 10.1109/TVCG.2007.70515.