# IATK: An Immersive Analytics Toolkit

Maxime Cordeil, Andrew Cunningham, Benjamin Bach, Christophe Hurter, Bruce Thomas, Kim Marriott, Tim Dwyer

# IATK: An Immersive Analytics Toolkit

Maxime Cordeil [*] Monash University
Andrew Cunningham [†] University of South Australia
Benjamin Bach [‡] Edinburgh University
Christophe Hurter [§] ENAC
Bruce H. Thomas [¶] University of South Australia
Kim Marriott [‖] Monash University
Tim Dwyer [**] Monash University

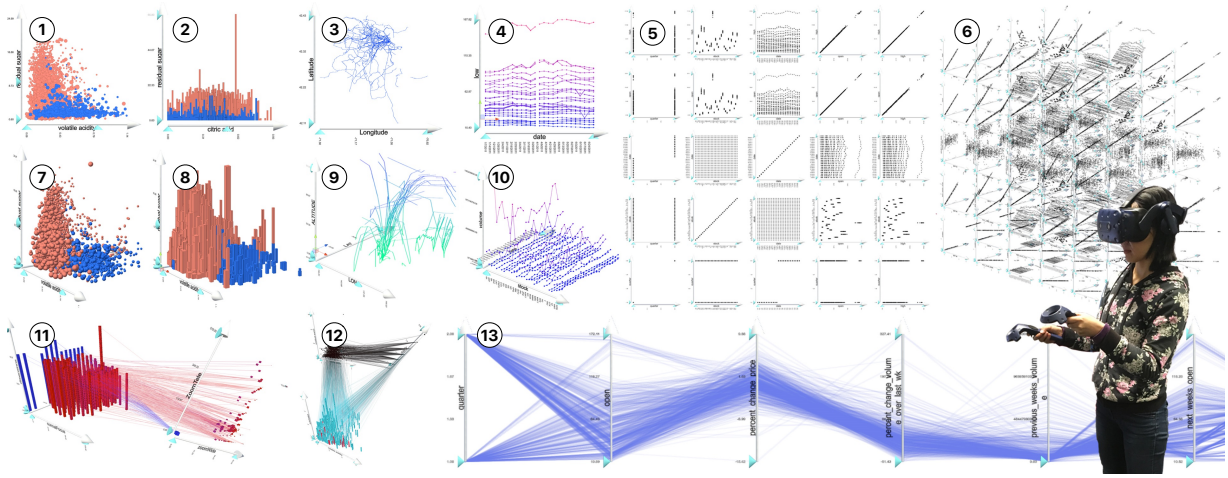

Figure 1: IATK is a scalable and expressive visualisaton toolkit that enables the creation of large, multidimensional data visualisations in immersive environments

## Abstract

We introduce IATK, the Immersive Analytics Toolkit, a software package for Unity that allows interactive authoring and exploration of data visualisation in immersive environments. The design of IATK was informed by interdisciplinary expert-collaborations as well as visual analytics applications and iterative refinement over several years. IATK allows for easy assembly of visualisations through a grammar of graphics that a user can configure in a GUI—in addition to a dedicated visualisation API that supports the creation of novel immersive visualisation designs and interactions. IATK is designed with scalability in mind, allowing visualisation and fluid responsive interactions in the order of several million points at a usable frame rate. This paper outlines our design requirements, IATK's framework design and technical features, its user interface, as well as application examples.

**Index Terms:** Human-centered computing—Visualization—Visualization techniques—; Human-centered computing—Visualization—visualisation design and evaluation methods

[*]maxime.cordeil@gmail.com
[†]Andrew.Cunningham@unisa.edu.au
[‡]bbach@inf.ed.ac.uk
[§]christophe.hurter@enac.fr
[¶]bruce.thomas@unisa.edu.au
[‖]kim.marriott@monash.edu
[**]tim.dwyer@monash.edu

## 1 Introduction

This paper introduces IATK, an open-source toolkit[1] that enables users to create data visualisations and immersive analytics systems. Hardware and software development toolkits have significantly contributed to the development of the fields of virtual [26] and augmented [37] reality, computer graphics[2], and data visualisation [10]. IATK has been designed to provide an expressive framework for building immersive data visualisations that scale to large datasets. As a toolkit, IATK provides both a high-level GUI interface for simple authoring and a low-level API for fine control and extending the visualisations.

Toolkits for 3D immersive environments (MR/VR/AR) have a long tradition [7, 37], e.g. facilitating object tracking [8, 50] or 3D modeling and rendering [48, 49][3]. Most recently, Unity has become a standard platform for developing immersive environment. Such toolkits form the backbone of development, education, and dissemination of computing software technology by providing simple access to routines and standard solutions, as well as facilitating development and prototyping.

An application area of immersive environments that is gaining increasing attention is data visualisation in immersive environments, commonly called *immersive analytics* [39]. This development is driven by the desire to investigate new means to explore large and potentially complex datasets using new and emerging display and interaction technologies. While data visualisation has traditionally focused on 2D media such as screen and paper, a range of novel applications are exploiting stereoscopic perception for the visualisation of 3D spatial datasets such as anatomic data [22], abstract high-dimensional data [17], and networks [18, 35]. Direct and natural interaction in immersive environments can support selection

[1]github.com/MaximeCordeil/IATK
[2]OpenGL: opengl.org
[3]OpenSceneGraph: openscenegraph.org.

and manipulation of 3D content [4]. Eventually, data visualisations can be embedded into the real world to support in-situ decision making [5, 54]. As the technology continues to improve and become more affordable, we expect an increase in both research on novel visualisation techniques in immersive environments and a wider range of applications for immersive visualisation [2, 3]. To cope with this growing demand and to increase exploration in the field, immersive analytics requires its own set of toolkits, providing standard routines, predefined visualisations and support for creating novel visualisations and interactions.

IATK addresses this gap. It is inspired by existing toolkits and concepts for visualisation on traditional 2D media such as screen and paper (notably Vega-lite [46] and The Grammar of Graphics [53]). Although there are existing immersive visualisation toolkits (DXR [47], VR-viz [45], and Glance [24]), IATK has been designed with a focus on scalability, data visualisation design, and user interaction. Additionally, IATK has emerged from several years of development of visualisation applications [17, 31] in collaboration with Monash University, the University of South Australia, ENAC and Edinburgh University. IATK builds upon previous work with the following contributions:

- A **composable grammar** of visualisation primitives.
- A **high-level interface** for rapidly constructing advanced interactive immersive visualisations in the Unity editor.
- **Rendering millions of items**, allowing IATK to be used in actual real-world scenarios.
- An extensive set of **example visualisations** e.g. scatterplots, scatterplot matrices, line graphs, parallel coordinates plots, trajectories and bar charts.
- An **expressive API** for building novel immersive visualisations.
- **Extended interaction capabilities** for brushing and linking, spatial interaction, and direct manipulation.

In the remainder of this paper, we outline existing toolkits for both immersive applications in general and visualisations in particular (Section 2). We then define four design requirements for IATK (Section 3) and explain which features in IATK meet these criteria. We introduce the IATK framework (Section 4). Further, we show how the concepts and components implemented from the design requirements of IATK (Section 5) empower users to build reactive, advanced immersive visualisations that enable novel ways to explore data. For example, it simplifies building an augmented reality desktop setup that uses a head-mounted AR device and hand gestures to explore data. In Section 7 we provide details on how to explicitly build this setup and provide a series of use cases that demonstrates the power of IATK. Additionally we report on IATK's performance and provide feedback collected from Immersive Analytics researcher attendees at two recent international tutorials.

## 2 Related work

### 2.1 Visualisation in Immersive Environments

Immersive visualisation has been defined as *"the use of engaging, multisensory, embodied data and analysis tools to support exploration, communication, reasoning and decision making"* [15, 39]. Previous VR research for data visualisation has focused on CAVE environments for scientific visualisation [1, 19] and abstract information visualisation [42]. However, more recent work has shown the potential of head-mounted displays (HMDs) for perception and interaction [4] and collaboration [17]. HMDs have been used to explore networks [35], immersive ego and exocentric maps [56], origin-destination flow maps [55], trajectories and brain fibers [31]. They also offer potential for integrating visualisation into a real-world context by projecting data visualisations into a user's physical environment [5, 21, 54].

### 2.2 Information visualisation Toolkits

The formalisation of the grammar of graphics [6, 53] and research to automate the design of visualisations [38] have significantly contributed to the design processes of data visualisation. These formalisms provide a systematic approach to creating visualisations and usually include (i) *visual marks*, such as points, lines, and areas representing data objects, (ii) *visual variables* to change the appearance of marks (color, transparency, size, texture, etc.) [6], and (iii) a range of *visual layouts* to position marks (e.g., cartesian and polar coordinate systems, network layout, etc.) as well as a range of supporting elements such as labels, titles, grids and different scales (e.g., linear, log scale). Data is mapped to visual marks and their attributes in a processes called *visual mapping*. Combining these aspects in a grammar provides a systematic approach to describe a very large design space.

A wide range of authoring tools and visualisation toolkits are based on such abstractions. For example, Fekete [23] introduced the InfoVis toolkit, the first visualisation toolkit for Java Swing applications. The Infovis toolkit allows users to define visualisations from existing templates, control visual variable mappings and define user interactions. A similar toolkit is Prefuse [28]. More recently, D3 [10] was introduced to facilitate the design of web visualisations by providing a low-level toolkit for interactive graphics. Most recently tools have emerged to provide high-level declarative grammars for creating visualisations [9, 36, 43, 46]. Instead of coding, visual mappings and layouts are assigne in a declarative way, allowing for very rapid and automatic design.

### 2.3 Immersive Applications and visualisation toolkits

Few toolkits exist that bring visualisation concepts to virtual and augmented reality. The general concepts for data visualisation remain the same (visual marks, attributes, layouts, etc.), but immersive spaces involve a greater range of technologies and programming environments. For example, dedicated toolkits exist for many specific tasks: Vuforia for tracking[4], A-frame for building AR applications on the web[5], and ARKit for iOS[6].

For scientific visualisation, ParaView [1] and VTK [41] have been used to create high-quality large-volume isosurface, and flow data visualisation. While successful in medical and engineering applications, they do not facilitate the simple creation of information visualisations such as scatterplots, matrices, and trajectories. Glance [24] is a graphics-optimized toolkit for information visualisation, but it requires low-level programming instead of providing a declarative grammer like IATK. GUIs are provided in both iViz [20] and Visualitics[7]. However, iViz is a commercial platform for immersive visualisation and does not provide programming capabilities nor the range of visualisations possible with IATK. There are two immersive toolkits for the web, VRIA [13] and VR-Viz[8] that use javascript to program a predefined set of visualisations. VR-Vis is based on A-Frame and uses a grammar approach inspired by Vega-lite.

Most similar to IATK is *DxR* [47], an authoring toolkit based on Unity. It targets three levels of user expertise (beginner, intermediate, advanced) and the grammar-based approach of Vega-Lite [46]. DxR is designed for rapid prototyping, rather than scaling to large datasets. Moreover, IATK's grammar is more expressive than those of existing toolkits and which allows for the creation of additional visualisations such as parallel coordinates or multiple views. Finally IATK is meant to support large datasets for research but also for real-world applications, while remaining an open-source toolkit.

[4] https://www.vuforia.com
[5] https://aframe.io
[6] https://developer.apple.com/arkit
[7] http://www.virtualitics.com
[8] https://www.npmjs.com/package/vr-viz

## 3 DESIGN REQUIREMENTS

To inform the design of IATK, we established a set of design requirements. We gathered these requirements based on our own experience as data visualisation researchers but also with the feedback of visualisation practitioners willing to explore immersive interactive data visualisation. These practitioners are academics from core AR/VR and visualisation areas.

**R1—Expressiveness:** Exploring complex data requires a variety of visualisations (e.g. scatterplots, parallel coordinates, scatterplot matrices, graphs etc.), each of which providing a different and ideally complementary perspective on the data. Moreover, different aspects of the data (time, geography, connections, etc.), different data types (dense-sparse, small-large, networks, temporal, etc.), and a range of analytical tasks (finding clusters and outliers, estimating trends, etc.) call for their own visualisation solution. To increase the range of applications, our grammar and toolkit must allow for the design and configuration of a large variety of visualisations.

**R2—Simple authoring:** A range of visualisation practitioners with diverse backgrounds must be able to create immersive visualisations. Those users include researchers, students, visual designers, data engineers and data scientists. Simple authoring through GUIs is essential to help a wide range of users to rapidly and efficiently build data visualisations.

**R3—Scalability:** Modern datasets often have many data points and attribute dimensions. In VR/AR, the display space that surrounds the user requires a new approach to visualising big data. Further, fluid and responsive rendering and interaction is crucial for usability [32, 33].

**R4—Integration of MR with the desktop for authoring and exploration:** While MR provides new opportunities for visualising and interacting with data, traditional desktop environments (keyboard, mouse and screen) are still the defacto way to code and perform traditional authoring tasks. Bridging the two environments—desktop and MR—is essential to fully support efficient data visualisation design and exploration.

## 4 IATK FRAMEWORK

The design of the IATK framework is grounded in fundamental interactive visualisation models and principles. In particular the IATK framework implements:

- an abstracted **data model** based on tabular data;
- a simple **grammar of graphics** that focuses on geometry and aesthetic design of data visualisations [53];
- an advanced **interaction model** that supports filtering, brushing and linking, and drill-down operations to get details on demand.

We will explain each of these points in detail in the following.

### 4.1 Data model

IATK features a data model that abstracts data files into a data table used by the visualisation components. The tabular data contains observations (row) and named attributes (columns). Several attribute types are supported: numerical, date and time, and categorical. These attributes are normalised into float values in the range [0,1], which is optimal for GPU shader code (as described in Section 6), while functions are provided for converting from normalised values back to their original format. The data model contains an abstract `DataSource` class and an implementation `CSVDataSource` that supports comma separated values (CSV). This implementation covers a large proportion of datasets, either natively or through conversion tools. After reading the header of the CSV, `CSVDataSource` will attempt to automatically infer the datatypes of attributes within the data, however users are able to pass in a metadata file that specifies the types and ranges of specific attributes.

### 4.2 IATK grammar of graphics

A strong motivation behind IATK is to empower users with an expressive interface that allows them to create a wide variety of data visualisations. Rather than simply providing a disparate collection of predefined graphics, we chose to use a "grammar of graphics" approach [52, 53]. This captures the underlying commonalities between different kinds of statistical graphics and provides the user with the power to move beyond named graphics [52].

IATK is based on a simple grammar of graphics that is expressive enough to handle a wide variety of common data graphics. A simple visualisation is created by specifying three components: a view-frame, the geometric objects to be placed in the frame, and the mapping from data attributes to the spatial and visual attributes of the geometric objects [6, 51]. More complex visualisations can be built from simple visualisation by using the *compound view component*, which generates view-frames and the *linking visualisation component* which can be used to display visual links (i.e. lines) between data points of two visualisations.

**Simple and Compound View-Frames:** Abstract data visualisations use a variety of coordinate frames, such as 2D and 3D orthogonal axis, radial, the many kinds of map projections, and parallel axes. The grammar of IATK contains a **View-frame Type** element that specifies the choice of coordinate frame. Currently **1D**, **2D** and **3D orthogonal axes** are provided. View-frames can be combined into more complex objects using **Compound-view** elements. Currently **1D**, **2D** and **3D orthogonal grids** are provided.

**Geometric Objects and Visual Encoding:** The **Geometry** specifies the geometric objects to encode the data. The currently available geometries of IATK are *Points, Lines and Bars*. **Glyphs** consist of a set of predefined graphical elements to draw the geometries. *Points*, *Spheres*, *Quads* and *Cubes* are available to draw Point geometries, *Lines*, *Connected Dots and Lines* to draw Line geometries and Cuboids to draw Bar geometries. The **Colour** channel maps a set of colours to a data dimension. If the data dimension is *undefined*, the same colour applies to all the data points; a continuous data dimension is mapped to the colour channel, the user can define a colour gradient. The user can define a colour palette when a categorical data dimension is mapped to the colour channel. The **Size** channel maps a data dimension to the size of the glyph. If the data dimension is *undefined*, a default, customisable size is applied to all data points. A mapping to a data dimension allows the user to define a *minimum* and a *maximum* range. Users can control pixel-level features such as transparency and blending modes. This type of interaction enables information emergence in dense datasets, and reveals structural information in data. Transparency control can also be used as an additional channel to filter out pixels.

**Visual Linking:** Visual links [16] help users to visually connect the same data points across multiple visualisations and to understand relationships in data. Typically, visual links are lines that connect the same data points across multiple visualisations. In abstract data visualisation, e.g. on dashboards, visual links can connect a number of heterogeneous visualisations such as scatterplots, graphs and barcharts. Since the architecture of IATK is fluid and reactive, the design and orientation of the visual links between two visualisations are updated when the aesthetic design (e.g. axes values or colors) or the position and orientation of a visualisation is changed. Visual links can also be brushed if the data point is brushed (i.e. the line of the visual link visualisation corresponding to the brushed data point is highlighted with the brush color (see Brushing and Linking, Section 5). One consequence of this architecture is that IATK can be used to create a system like ImAxes [17], in which visualisations can be linked (Figure 1 (11,12)). Those links can be manipulated with hand controllers, e.g. to improve the perception of tclutter.

**Summary:** The IATK grammar enables the design and creation of a wide range of data visualisations commonly used in information visualisation and visual analytics, such as statistical charts (scatter-

plots and bar charts), trajectories, graphs and line charts (see Figure 1). However, this expressiveness means that it can be at first daunting to create a graphic using the full grammar. To simplify the creation of common data visualisations in IATK, we provide a number of predefined combinations of the above elements:

- **Scatterplot:** 2D and and 3D orthogonal axes with point geometric object (Figure 1 (1,7))
- **Scatterplot matrix:** a 2D or 3D grid of 2D or 3D scatterplots (Figure 1 (5,6))
- **Parallel coordinates:** a 1D grid of 1D axes with line geometry. (Figure 1 (13))

### 4.3 Interactive visualisation model

Efficient visual data exploration requires a set of interactions that include filtering data, making queries, select data points and changing the visual design to seek outliers, trends and more general structural information in the data [30]. IATK's interaction model is composed of manipulating the grammar of graphics using high-level interfaces (see next Section), and immersive interactions with the visualisations. This interaction model is integrated directly within the visualisation components and scales to support large datasets.

**Filtering:** Filtering allows a user to define useful data ranges on the visualisation's axes, filter out outliers and categories or ranges of data from other dimensions of the dataset. In IATK the user can specify a minimum and maximum value per axis. Changing these values results in re-scaling the range of a dimension. IATK also allows to filter out minimum and maximum values without re-scaling. The data points that are filtered out are not displayed and the filtered values on the axis are grayed out. Finally, users can build multidimensional queries to filter out data in additional dimensions. This is done by selecting one or several data dimensions and specifying minimum and maximum filter values for the selected dimensions.

**Brushing and linking:** In the process of creating several visualisations showing many facets of multidimensional data, users may need to relate the data points across visualisations. This is supported by interactive *brushing and linking* [11, 12, 34], an interactive process that enables the selection of data points in one visualisation and automatically highlights the same data points in other visualisations. A highly responsive interface [40] is required to ensure a smooth Brushing an Linking interaction.

**Details on demand:** Retrieving values of selected information is an essential task during the visual exploration or presentation of data. We defined an inverse projection function that transforms screen and model coordinates back to the normalized data space. This allows retrieving values on axes, or from 2D/3D pointing in a visualisation.

**Animated transitions and animations:** Changing the design of a data visualisation can alter the focus of a user. For example, if the geometry or data bindings are changed abruptly the user may lose track of the set of data points they were inspecting. Smooth animated transitions between visualisation designs are required to help users preserve focus and keep track of graphical objects of interest [29]. Animations based on temporal attributes help to understand the dynamics of data [44]. The IATK framework contains both animated transitions and attribute-based animation mechanisms.

## 5 IATK Interface

We implemented the IATK framework into Unity, as it is a broad immersive environment technology enabler that is accessible from beginners to expert designers and programmers. In consideration of requirement R2 (simple authoring), we provide two user-facing interfaces for working with the toolkit: a Graphical User Interface (IATK GUI) that integrates with the Unity editor and a direct Application Programming Interface (IATK API).

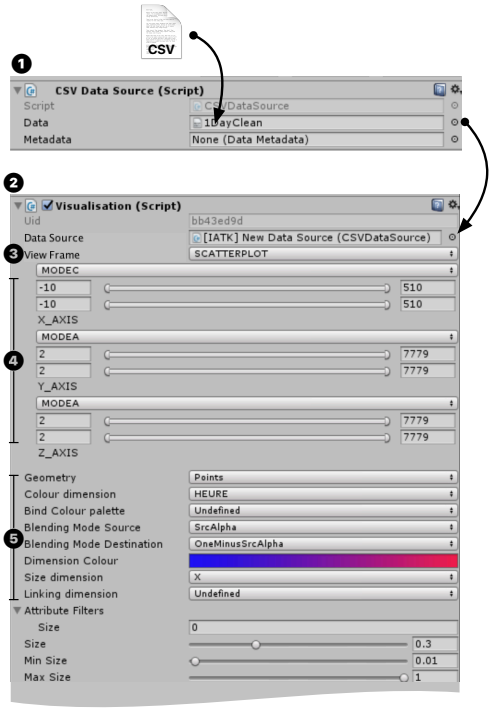


Figure 2: The IATK Data Source and Visualisation GUI panels.

### 5.1 IATK GUI

The Unity editor is composed of four main panels: the rendering window, the scene hierarchy that contains the rendered objects, a project browser and an inspector GUI. IATK features a set of visualisation authoring GUI panels that integrate directly within the Unity editor's inspector panel. By integrating with the Unity editor, IATK can leverage the WIMP interactions of the interface to bind data files and objects to other components, e.g. using drag and drop. This drag and drop interaction approach encourages rapid development and exploration of the visualisation design space. The elements of IATK GUI (see Figure 2) are:

The ① **IATK data GUI panel** contains a CSV file field, a graph data file field, and a meta data file field. A user drags and drops text files from the Unity project browser into the fields to configure the underlying datasource object.

The ② **IATK visualisation Editor panel** is the central GUI element component of IATK. It is enabled after creation once a user drags and drops a Data component, The ③ `visualisation Type` dropdown menu selects the type of visualisation to create: either a scatterplot, a parallel coordinates or a scatterplot matrix visualisation. The parameters unique to each visualisation appear in the ④ visualisation panel.

**Scatterplot item:** when the scatterplot item menu is selected, the visualisation Editor panel displays three groups of controls (referred to as *attribute filters*), each consisting of an attribute dropdown and two range sliders. The three attribute filters correspond to the X, Y, and Z axes of the visualisation. When the user selects a new attribute from a dropdown list, the visualisation axis animates to the new attribute values. The two range sliders for an attribute filter normalise and filter values in the axis respectively.

**Scatterplot Matrix item:** when selected, the visualisation Editor panel displays three editable lists of attribute filters. The three lists correspond to the X, Y, and Z dimensions of the SPLOM.

**Parallel Coordinates item:** when selected, the visualisation Editor panel displays a reorderable list of attribute filters that represent

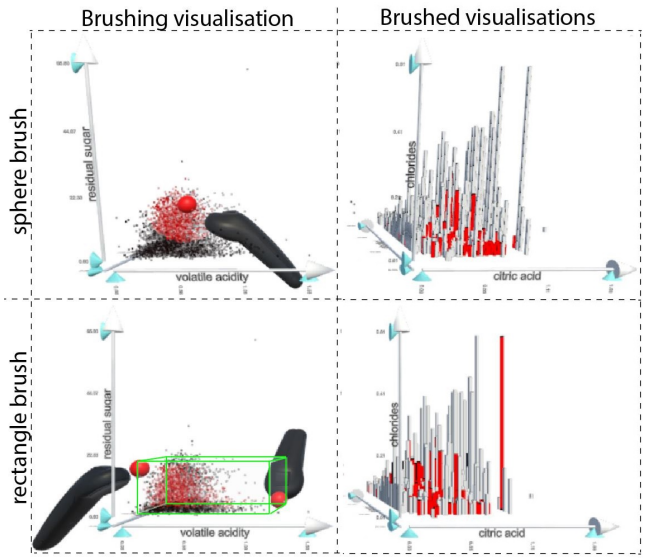


Figure 3: Sphere and box Brushing and Linking in IATK – a selection is made in a *Brushing visualisation* and the selected points are highlighted in the *Brushed visualisations* (red marks)

the axes of the visualised PCP.

Aesthetics are defined in the ⑤ aesthetic panel. The geometry of the visualisation is defined with a dropdown menu that contains the geometries available in IATK's grammar. General aesthetics (e.g. color, size) are also defined by selecting attributes from dropdown menus.

**IATK Linked visualisations** enables visual links between visualisations. The user drags and drops visualisation objects from the hierarchy onto the visualisation fields of the Linked visualisation GUI. A boolean value enables and disables the linked lines.

**IATK Brushing and Linking** propagates a data point selection from a *Brushing visualisation* to a set of *Brushed visualisations* (Figure 3). The user can define the geometry (**Brush Type**), the input parameters (**Input1** and **Input2**) and the aesthetics (i.e. the color and size of selected data points) to create a custom selection. The Input1 and Input2 fields are generic as they only require a *Transform* component (i.e. the position and rotation of an object in the scene). Hence the *Transform* of any object in the scene can be used to specify the selection. IATK contains two **Brush Type** elements that allow 1D, 2D and 3D selections:

- *SPHERE* requires a *Transform* component to be attached to Input1 and a radius value in the normalized range [0,1]. This Brush Type allows a disc selection on a 2D visualisation (i.e. the intersection of the sphere on the 2D plane) and when the radius is set to a small value, allows single point brushing. In 3D, the sphere brushes all the 3D points within its radius.
- *BOX* requires two object *Transform* components to be attached to Input1 and Input2. This Brush Type enables bi-manual interaction to draw a box selection in 3D, or a rectangle in 2D.

The Brushing and Linking component also exposes a list of the selected data points' `ids` that enables the user to obtain details on demands of the selected data points.

Interactions with the GUI are fluid and responsive up to 1M points (see Section Performance), i.e. when a visualisation parameter is changed, the visualisation in the Unity editor rendering window or the headset view is updated instantly. This ensures an uninterrupted flow of actions [14] to explore data and build visualisations.

### 5.2 IATK low-level API

The GUI editor allows users to create a wide range of data visualisations with different designs. However, if a user wants to create a custom visualisation outside the constraints of the GUI, an API is required to enable more expressiveness. This supports more flexibility, modularity, and fine grained control over the design of an immersive visualisation. IATK provides a C# and Javascript API to address this. The API is designed around a fluent interface, which encourages a domain specific language style approach to the interface design [25]. This aspect of fluent design supports an interface that reflects the grammar of graphics defined by IATK. We follow the conventions of D3, where a visualisation object is created by chaining design commands to a declared object. In IATK, a `ViewBuilder` is declared by defining the base geometry and the name of the visualisation. Aesthetics commands (e.g. position values, color, size etc.) are chained to the `ViewBuilder`. When applied the builder returns a `View` object. To illustrate, the following code outlines the calls required for a basic 3D point visualisation:

```
int number_of_points;
float[] data_array_x, data_array_y, data_array_z, data_array_size;
Color[] data_array_color;
// create a view builder with the point topology
ViewBuilder vb = new ViewBuilder (geometry, "visualisation name").
        initialiseDataView(number_of_points).
        setDataDimension(data_array_x, ViewBuilder.VIEW_DIMENSION.X).
        setDataDimension(data_array_y, ViewBuilder.VIEW_DIMENSION.Y).
        setDataDimension(data_array_z, ViewBuilder.VIEW_DIMENSION.Z).
        setSize(data_array_size). // set the size of each data point
        setColors(data_array_color); // set the color of each data point
View view = vb.updateView().apply(gameObject, nil);
```

Unity uses the *Mono* programming framework environment. Mono includes the *LINQ* framework to perform advanced data filtering and queries (e.g. using SQL-like commands to select subsets of data). Given this rich integrated programming environment, the API of IATK allows to specify interactive data visualisations with few lines of code (Section 7.3 details an example of use with LINQ).

## 6 Architecture and Performance

Several architectural considerations emerge from the requirements identified in Section 3. To support R3 (scalability), we designed IATK to leverage modern GPU processors where possible, in lieu of the higher level frameworks offered by Unity. To support R2 (simple authoring), we have designed a multi-level architecture that provides a high-level GUI driving a low-level API, both accessible to the user. To support R4 (MR and desktop integration), we implemented a serialisation facility that preserves the design changes at runtime and rebuilds the modified visualisations back into the editor. This architecture is modular and can be extended. New `Visualisation` types can be created that use the `ViewBuilder` and interaction model.

**Scalability:** The scalability (R3) of the IATK architecture revolves around the use of a data-driven mesh model. Naively rendering the data points as Unity game objects would limit performance beyond one-thousand data points. In IATK's data-driven mesh model, dataset attributes and aesthetic parameters are bound and encoded to individual vertices in a mesh object. Computationally expensive tasks are then handled by a custom set of *visualisation shader programs*, moving the computation from the CPU to the GPU where it can be parallelised.

Figure 4 illustrates the rendering pipeline for the sphere glyph. Each stage in the pipeline represents a data transformation. In the first CPU-bound stage, data attributes are mapped to IATK's grammar of graphics. The second stage encodes each datapoint into a unique vertex, mapping aesthetics such as size to the UV components of the vertex. Dimension manipulations, such as filtering and normalisation, occur on the GPU in the vertex shader stage. This is

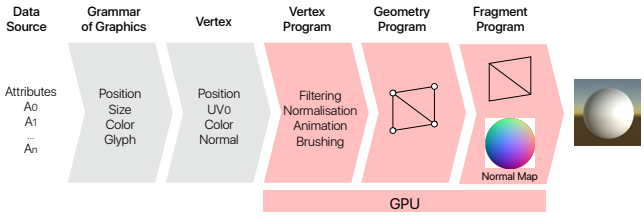


Figure 4: The rendering pipeline of IATK. Each stage represents a data transformation. Computational expensive stages, such as normalisation and animation, are performed on the GPU.

also the stage where animation is applied, interpolating positions and other aesthetics such as size. In the geometry stage, IATK creates geometry such as cubes and quads directly on the GPU, reducing communication from the CPU that forms a common bottleneck in 3D applications. The glyphs are rendered during the final fragment stage. For the sphere shader, a pre-computed normal map of a sphere is applied to a lighting model to give the quad the appearance of a physically lit sphere while being composed of far less actual vertices. In this way, the shader has rendered what appears to be a complex shape using minimal data passed from the CPU.

**Fluid interaction:** To achieve fluid interaction, we propose a fast implementation of Brushing and Linking that leverages the Compute Shader capabilities of the GPU. The computationally expensive aspect of brushing involves testing every data point of the visualisation against the brush shape (a $O(n)$ task). Moving this algorithm to the GPU provides several performance benefits. The algorithm can be parallelised, increasing performance by a factor of 32. The algorithms can leverage the float performance of the GPU. Finally, performing brushing on the GPU means the brush data is immediately available to the rendering pipeline without needing further CPU communication. The Brushing and Linking uses a *compute kernel*. The data attributes are passed to the kernel in a similar process to the vertex stage of the rendering pipeline. The kernel tests each data point against the given brush position and shape (either a box, cube, or sphere). Brushed data point IDs are written to a brushing "texture" that has a texel for each data point. The state of a data point (i.e. whether it is brushed or not) is encoded in a color channel in the texture. This texture is then passed to the vertex program stage of the rendering pipeline.

**Integration of desktop and MR:** Unity makes a distinction between editing and runtime. During editing the user sets up the environment and simulations. During runtime, the simulations play out and the user can experience the application. While editing is supported during runtime, any changes are reset when runtime is finished. For a game engine, this behaviour is ideal. For a visualisation designer, this behaviour is detrimental to the exploratory process. We implemented a serialisation facility that preserves the design changes at runtime and rebuilds the modified visualisations back into the editor. This serialisation supports continuity between the desktop and MR environments (R4).

### 6.1 Performance graph

We profiled IATK with the Oculus Rift CV1, the Meta 2, and the HoloLens to determine performance with large datasets. The Oculus Rift CV1 and Meta 2 were running on an Intel(R) Core(TM) i7-7800X CPU (3.5Ghz), 32GB of RAM and an Nvidia Geforce GTX1080 graphics card (8GB VRAM). We generated six 3D scatterplot datasets of increasing size (10,000, 100,000, 1M, 2M, 5M and 10M data points). The data point positions were randomly generated. The scatterplot was positioned such that it filled the device's view.

Figure 5 summarises the results of the benchmark. Up to 2M points the refresh rate constantly hits 90 FPS on the Oculus Rift, which then decays linearly to 25 FPS when tested under the extreme conditions of 10M points. Similarly the Meta 2 maintains 100 FPS, dropping slightly by 2M points, and then linearly dropping. It is worth noting that these framerates up to 2M are artificially capped by the hardware. The HoloLens ran at an optimal 60 FPS at 1,000 data points. However, the hardware (CPU, GPU and RAM) in the HoloLens are more akin to mobile class hardware. As such, the performance dropped linearly from 1,000 points. Subjectively, it was still usable at 10,000 points (running at 41 FPS), but by 1M points the system is unusable. We noted that when fragment shader overdrawing occurs, the performances of the toolkit are degraded. It is then necessary to adjust the size of the geometry to minimise overdrawing. Finally the responsiveness of the Unity GUI editor scripts decreases dramatically from 5M data points, with a response time approaching 5 seconds (see Figure 5).

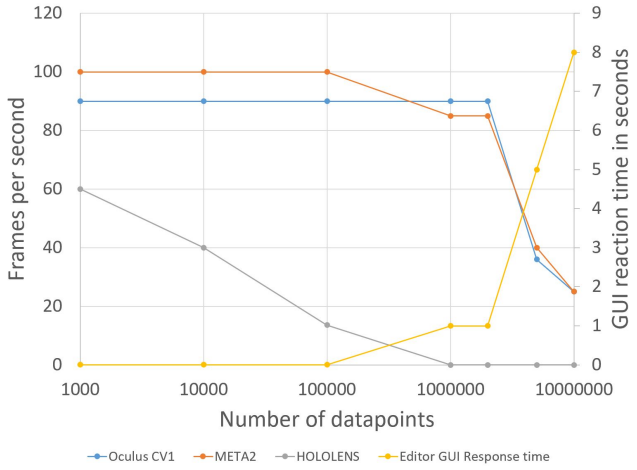


Figure 5: IATK FPS and editor response time performances

## 7 Scenarios and Use Cases

We demonstrate how IATK can be used to build a range of interactive immersive visualisation applications within the Unity Editor GUI and using the IATK API. We illustrate the use of our toolkit with three scenarios and then demonstrate how to partially clone existing immersive visualisation techniques. We close each scenario with a high level conclusion in italics.

### 7.1 Interactive VR setup with IATK GUI

Virtual Reality immersive environments are a particularly interesting medium for investigating high-dimensional and multivariate data because they offer spatial real estate all around the user to display and arrange a large number of visualisations. IATK *can be used to setup this environment without having to write a single line of code*. We illustrate building such an environment with IATK using a scenario that involves Alice, a data scientist with beginner knowledge of Unity. Alice's goal is to setup a VR dashboard to explore a Dow Jones index data set. Alice requires three key components to explore the data: a scatterplot matrix, a 3D scatterplot and parallel coordinates. Alice creates a data source and the corresponding three visualisation types using the GUI contextual menu in the scene hierarchy of Unity. After exploring different data mapping and aesthetic options for all the visualisations (again using the IATK GUI), she arranges them into the virtual environment using the Unity scene view. Further, Alice creates a `BrushingAndLinking` component to create a brushing interaction that links the three views. Alice has now created a setup in which she can interact with the 3D line chart that shows the evolution of index volumes and inspect specific periods by selecting

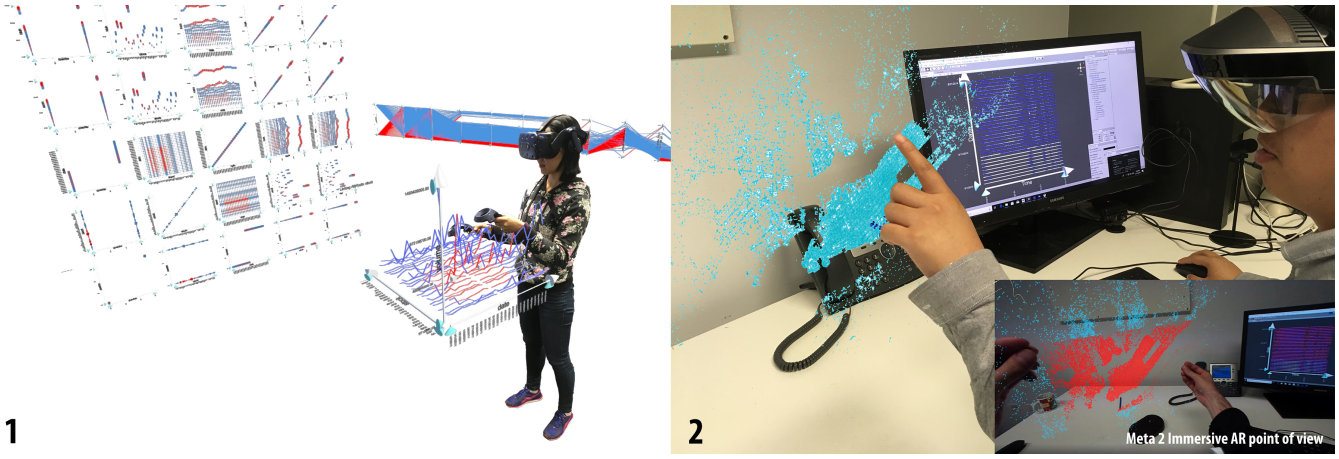


Figure 6: Two examples of immersive visualisation with IATK1) An immersive VR visualisation platform to analyze multidimensional data (Dow Jones dataset). The user brushes a 3D trendline visualisation and the points are highlighted in red in both the scatterplot matrix and the parallel coordinates. 2) An augmented desktop visualisation platform with the Meta 2. The user interacts with a map displayed in AR above the physical desktop, and the selected data points highlight on the 2D visualisation on screen in Unity.

sets of points in the visualisation (Figure 7(1)). The corresponding data is highlighted on the virtual scatterplot matrix wall and parallel coordinates. With this setup, she benefits from a room-size display environment in which she can get more visual detail from the data she inspects, and leverage bi-manual input to quickly select data points of interest in a visualisation.

*This scenario showed how IATK facilitated the creation (R2) of advanced, highly interactive multidimensional data visualisation in VR. Using only GUI actions, the system is expressive enough to build an interactive dashboard without programming.*

### 7.2 AR-desktop exploration of geospatial data

Multivariate geospatial data is analysed in a wide variety of domains and industries. The data consists of latitude and longitude values with additional fields. The exploration of this data often consists of mapping the data points to 2D glyphs (e.g. position and size for certain data dimensions). In this example scenario we are interested in Tom, an analyst working for a famous ridesharing company. Tom has a very large open source dataset[9] of pickup locations in New-York that also contains the date and time of the day and other data dimensions. Tom's aim is to analyse the correlation between the date, the time of the day and the pickup locations. Tom uses the IATK GUI to design a 2D (Date,Time) scatterplot visualisation of his data. Realising that the data was incomplete, Tom uses his traditional desktop-based data tools to clean and wrangle the data. Returning to Unity, Tom creates a geographic 2D map (latitude, longitude) with the updated data. Tom then declares a brushing and linking mechanism, and attaches the hands *Transforms* provided by the AR Meta 2 Unity framework. Tom runs the app and places the map to his left, as if it was a 2.5D holographic screen (Figure 7 (2)). In order to maximise the number of visible datapoints, Tom re-sises the visualisation object. Using hand interactions, Tom selects single and multiple datapoints which are then highlighted on the desktop visualisation.

*Because IATK offers continuity between immersive and desktop environments (R4), a user can leverage familiar desktop tools while visualising their data in AR. Tom can keep using his desktop applications with his mouse and keyboard, but also benefits from direct manipulation interaction on the immersive map, thanks to the integration with a see-through device with hand-tracking capabilities.*

[9] www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city

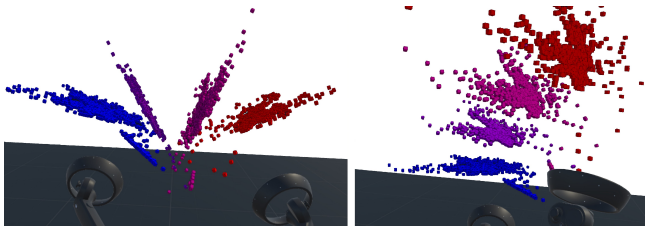

Figure 7: A fanning interaction for a 3D layered visualisation. Layers of a large dataset are "fanned" out between the user's hands.

### 7.3 "One million data points in my hands"

The input modalities and affordances in immersive environments are of particular interest for exploratory immersive analytics research [39]. For example 6DOF VR controllers offer direct interaction with data visualisations. However the integration of these inputs (e.g. position of motion-tracked controllers) with visualisation systems is not easy and often requires fastidious programming. The API of IATK is a powerful yet concise programming tool to easily create novel interactions that take advantage of these input modalities. Here we show how we can leverage Unity and its programming environment to create a fanning interaction [27] of an exploded 3D visualisation using two motion tracked controllers.

A 3D layered visualisation of latitude, longitude and base (same dataset used in the previous scenario) can be obtained with the following code. In order to create a fanning interaction it is necessary to isolate each layer into a single IATK View. IATK's compatibility with the integrated query framework LINQ allows us to perform complex data filtering to effectively isolate each base into separate IATK Views layers. First, we create a filter function that uses LINQ to filter IATK's `DataSource` object:

```
// filters the array on a particular value in another dimension
Filter baseFilter = (array, datasource, value, dimension) => {
    return array.Select((_base, index) => new { _base, index })
                .Where(b => datasource.getOriginalValuePrecise(
    array[dimension].Data[b.index],dimension).ToString() == value)
                .Select(b => b._base).ToArray(); };
```

This function is used to facet the data into layers when building the views. We then add the following code to the update loop to

interpolate the layer positions, creating a fluid interaction:

```
// interpolate the position and orientation of each facet
layers[i].transform.position = Vector3.Lerp(controllerLeft.position,
        controllerRight.position, (float)i / (float)(layers.Length − 1));
layers[i].transform.rotation = Quaternion.Lerp(controllerLeft.rotation,
        controllerRight.rotation, (float)i / (float)(layers.Length − 1));
```

The resulting system from these few lines of code produces a fluid fanning interaction that interpolates the positions and rotations of the layers between the two controllers *Transforms* (Figure 7). The user is able to "fan out" their hands and see the faceted data spreadout fluidly.

*This scenario demonstrates how to quickly build a fanning technique with few lines of code and a high level LINQ syntax. This technique can be further studied e.g. by extending the interpolation method between the layers to parametric curves such as Beziers curves.*

### 7.4 Reproducing immersive visualisation applications

In order to further demonstrate IATK expressivity we show how previous immersive visualisation interaction principles and systems can be implemented with IATK.

**Minimal implementation of ImAxes** ImAxes [17] allows users to place embodied data axes in 3D space and displays visualisations according to the relative placement of those axes, as determined by a spatial grammar. This grammar recognises spatial layouts such as perpendicular or parallel axes to produce, respectively, scatterplots and parallel coordinates. Additionally links appear between visualisations when they are within a distance threshold. IATK can be used to recreate all the defined visualisations (e.g. scatterplots, PCPs, links) from ImAxes. The grammar recogniser of ImAxes [10] can be directly reused with the IATK framework. Furthermore, IATK can be used to create more visualisations and linked visualisations based on the type of data (e.g. continuous vs discrete). For example, ImAxes can be extended with linked bar charts and scatterplots (see Figure 1 (11)).

**FiberClay selection style** FiberClay [31] is a VR aircraft trajectory visualisation tool that implements a specific bi-manual interaction to select a set of trajectories (i.e. a trajectory is selected if it intersects the two controllers). First, IATK can be easily used to author a 3D immersive aircraft trajectory visualisation using simple GUI operations within the Unity editor (see Figure 1 (9)). Second, the flexibility of IATK allows a programmer to alter the *Compute Shader* code that handles the selection of graphical objects. Since the framework already takes into account two input *Transforms* to select data, it is sufficient to store the index of the selected trajectory in another color channel of the brushing texture and only select a line when the two inputs intersect it.

**Immersive AR** Recently Bach et al. [4] studied immersive augmented reality visualisation with multiple data selection tasks (see section 2). The study used the HoloLens, which has usability issues such as a narrow field of view and the results may have depended on limitations of the device itself. IATK facilitates the work of researchers to reproduce the same conditions in a device agnostic environment. In this way, it opens possibilities for research to test devices with different properties to make potential new insights and contribute new results to the field of Immersive Analytics research.

*IATK allows researchers to recreate immersive techniques and systems that already exists, making it a support tool for research to reproduce previous experiments and to extend research to make new insights on Immersive Analytics usability.*

## 8 Discussion

In this section, we elaborate a discussion on the current limits and future extensions of our toolkit based on feedback collected during two IATK tutorials (IEEE VIS 2018 and IEEE BDVA 2018). The participants of the workshops were data visualisation, virtual reality and human-computer interaction experts (researchers, PhD students, and engineers from industry).

**Usability and community impact:** IATK has been particularly well received by the research community. In particular, one researcher was eager to further explore the concept of "flexible linked visualisation" introduced in ImAxes [17], and was able during the tutorial to assemble the basic interaction bricks required to emulate the system. Another researcher found the toolkit very useful as they wanted to explore visualisation perception in VR. In that case IATK offers the possibility to rapidly create different visualisation designs (e.g. spheres, plots, quads with different sizes and colors) with potentially large datasets to test users' perception. Finally during a workshop a researcher ported IATK to Linux. This feedback suggest a strong community interest for IATK.

**Expressiveness:** We received positive feedback regarding expressiveness with many participants reporting that they were able to create the visualisations they wanted. Considering the descriptive power of the toolkit, we have shown that IATK can reproduce a diverse range of existing system. Considering the generative power of IATK for new visualisations, some participants requested more geometries, glyphs and View Frames (e.g. radar plots, surfaces, node-link graphs). Since the architecture and model of IATK exposes a data-driven `Mesh` object that can render any primitive via the graphics card, it is possible to extend the grammar to include more visualisations, addressing these requests. In future work, more glyphs and View Frames will be available directly on the GUI to allow for more combinations and expressiveness.

**Scalability:** Based on performance measures and participants' feedback, the scalability of the tool for VR and thethered AR platforms covers a wide range of real-world dataset sizes. However, according to our measures, performances are still relatively low in mobile AR environments (e.g. on HoloLens which possesses a low performance GPU). For mobile AR devices with DirectX 11 support, we are currently exploring how we can optimize performances e.g. using GPU instancing for point geometry.

## 9 Conclusion

We contributed the design and implementation of IATK a scalable, expressive and device agnostic data visualisation toolkit for immersive environments. Informal feedback from our workshop tutorials suggests that IATK supports a simple yet expressive grammar of graphics that allows users from diverse backgrounds to create a wide range of visualisations and to explore new visualisations using the GUI and the API. It also defines a high-level interaction model that enable filtering, brushing and linking and details on demand. We demonstrated how our toolkit can be employed to create innovative data visualisation setups in VR and AR, and how it can be used to reproduce immersive data visualisation tools and techniques. The performance tests we ran indicate that our toolkit can be effectively used for immersive visual analytics of large data. Additionally, we gathered useful feedback from a community of Immersive Analytics researchers and plan to extend our grammar of graphics to allow for more expressiveness. More precisely, in future work we plan to add grammar operators to support visualisation composing and layout. Further investigation of the usability of the GUI interface (at the moment rather technical) would help to us to refine and streamline the authoring experience. In particular, there is the opportunity to take advantage of the live VR editing features available in current Unity beta versions to allow users to author visualisations without switching back and forth between headset and desktop.

## 10 Acknowledgements

This research was supported under the Australian Research Council's Discovery Projects funding scheme (#DP180100755). We also acknowledge Kingsley Stephens for his technical help.

[10] github.com/MaximeCordeil/ImAxes

## REFERENCES


[1] J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large data visualization.

[2] B. Bach, M. Cordeil, T. Dwyer, B. Lee, B. Saket, A. Endert, C. Collins, and S. Carpendale. Immersive analytics: Exploring future visualization and interaction technologies for data analytics.

[3] B. Bach, R. Dachselt, S. Carpendale, T. Dwyer, C. Collins, and B. Lee. Immersive analytics: Exploring future interaction and visualization technologies for data analytics. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces*, pp. 529–533. ACM, 2016.

[4] B. Bach, R. Sicat, J. Beyer, M. Cordeil, and H. Pfister. The hologram in my hand: How effective is interactive exploration of 3d visualizations in immersive tangible augmented reality? *IEEE Transactions on Visualization & Computer Graphics*, (1):1–1.

[5] B. Bach, R. Sicat, H. Pfister, and A. Quigley. Drawing into the ar-canvas: Designing embedded visualizations for augmented reality. In *Workshop on Immersive Analytics, IEEE Vis*, 2017.

[6] J. Bertin, W. J. Berg, and H. Wainer. *Semiology of graphics: diagrams, networks, maps*, vol. 1. University of Wisconsin press Madison, 1983.

[7] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. Vr juggler: A virtual platform for virtual reality application development. In *IEEE Virtual Reality*, pp. 89–96. IEEE, 2001.

[8] M. Billinghurst, J. Bowskill, M. Jessop, and J. Morphett. A wearable spatial conferencing space. In *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pp. 76–83. IEEE, 1998.

[9] M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *IEEE transactions on visualization and computer graphics*, 15(6), 2009.

[10] M. Bostock, V. Ogievetsky, and J. Heer. $D^3$ data-driven documents. *IEEE Transactions on Visualization & Computer Graphics*, (12):2301–2309, 2011.

[11] A. Buja, D. Cook, and D. F. Swayne. Interactive high-dimensional data visualization. *Journal of computational and graphical statistics*, 5(1):78–99, 1996.

[12] A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle. Interactive data visualization using focusing and linking. In *Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on*, pp. 156–163. IEEE, 1991.

[13] P. W. Butcher, N. W. John, and P. D. Ritsos. Towards a framework for immersive analytics on the web.

[14] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In *Proceedings of the SIGCHI Conference on Human factors in computing systems*, pp. 181–186. ACM, 1991.

[15] T. Chandler, M. Cordeil, T. Czauderna, T. Dwyer, J. Glowacki, C. Goncu, M. Klapperstueck, K. Klein, K. Marriott, F. Schreiber, et al. Immersive analytics. In *Big Data Visual Analytics (BDVA), 2015*, pp. 1–8. IEEE, 2015.

[16] C. Collins and S. Carpendale. Vislink: Revealing relationships amongst visualizations. *IEEE Trans. on Visualization and Computer Graphics (Proc. of the IEEE Conf. on Information Visualization (InfoVis))*, 13(6), Nov./Dec. 2007. doi: 10.1109/TVCG.2007.70521

[17] M. Cordeil, A. Cunningham, T. Dwyer, B. H. Thomas, and K. Marriott. Imaxes: Immersive axes as embodied affordances for interactive multivariate data visualisation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pp. 71–83. ACM, 2017.

[18] M. Cordeil, T. Dwyer, and C. Hurter. Immersive solutions for future air traffic control and management. In *Proceedings of the 2016 ACM Companion on Interactive Surfaces and Spaces*, pp. 25–31. ACM, 2016.

[19] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pp. 135–142. ACM, 1993.

[20] C. Donalek, S. G. Djorgovski, A. Cioc, A. Wang, J. Zhang, E. Lawler, S. Yeh, A. Mahabal, M. Graham, A. Drake, et al. Immersive and collaborative data visualization using virtual reality platforms. In *Big Data (Big Data), 2014 IEEE International Conference on*, pp. 609–614. IEEE, 2014.

[21] N. A. ElSayed, B. H. Thomas, R. T. Smith, K. Marriott, and J. Piantadosi. Using augmented reality to support situated analytics. In *Virtual Reality (VR), 2015 iEEE*, pp. 175–176. IEEE, 2015.

[22] J. Falah, S. Khan, T. Alfalah, S. F. M. Alfalah, W. Chan, D. K. Harrison, and V. Charissis. Virtual reality medical training system for anatomy education. In *2014 Science and Information Conference*. IEEE, aug 2014. doi: 10.1109/sai.2014.6918271

[23] J.-D. Fekete. The infovis toolkit. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pp. 167–174. IEEE, 2004.

[24] D. Filonik, T. Bednarz, M. Rittenbruch, and M. Foth. Glance–generalized geometric primitives and transformations for information visualization in ar/vr environments. In *Proceedings-VRCAI 2016: 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry*, vol. 1, pp. 461–468, 2016.

[25] M. Fowler. *Domain Specific Languages*. Addison-Wesley Professional, 1st ed., 2010.

[26] L. Gandel and J. Jomier. Tutorial 2: Developing virtual reality applications with the visualization toolkit (vtk). In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*. IEEE, 2017.

[27] L. He, A. Guayaquil-Sosa, and T. McGraw. Medical image atlas interaction in virtual reality.

[28] J. Heer, S. K. Card, and J. A. Landay. Prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 421–430. ACM, 2005.

[29] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE transactions on visualization and computer graphics*, 13(6):1240–1247, 2007.

[30] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. *Queue*, 10(2):30, 2012.

[31] C. Hurter, N. H. Riche, S. M. Drucker, M. Cordeil, R. Alligier, and R. Vuillemot. Fiberclay: Sculpting three dimensional trajectories to reveal structural insights. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018. doi: 10.1109/TVCG.2018.2865191

[32] D. A. Keim. Pixel-oriented visualization techniques for exploring very large data bases. *Journal of Computational and Graphical Statistics*, 5(1):58–77, 1996.

[33] D. A. Keim. Visual exploration of large data sets. *Communications of the ACM*, 44(8):38–44, 2001.

[34] P. Koytek, C. Perin, J. Vermeulen, E. André, and S. Carpendale. Mybrush: Brushing and linking with personal agency. *IEEE transactions on visualization and computer graphics*, 24(1):605–615, 2018.

[35] O.-H. Kwon, C. Muelder, K. Lee, and K.-L. Ma. A study of layout, rendering, and interaction methods for immersive graph visualization. *IEEE transactions on visualization and computer graphics*, 22(7):1802–1815, 2016.

[36] J. K. Li and K. Ma. P4: Portable parallel processing pipelines for interactive information visualization. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018. doi: 10.1109/TVCG.2018.2871139

[37] B. MacIntyre, M. Gandy, S. Dow, and J. D. Bolter. Dart: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pp. 197–206. ACM, 2004.

[38] J. Mackinlay. Automating the design of graphical presentations of relational information. *Acm Transactions On Graphics (Tog)*, 5(2):110–141, 1986.

[39] K. Marriott, F. Schreiber, T. Dwyer, K. Klein, N. H. Riche, T. Itoh, W. Stuerzlinger, and B. H. Thomas, eds. *Immersive Analytics*. Springer International Publishing, 2018. doi: 10.1007/978-3-030-01388-2

[40] R. B. Miller. Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pp. 267–277. ACM, 1968.

[41] P. O'Leary, S. Jhaveri, A. Chaudhary, W. Sherman, K. Martin, D. Lonie, E. Whiting, J. Money, and S. McKenzie. Enhancements to vtk enabling scientific visualization in immersive environments. In *Virtual Reality (VR), 2017 IEEE*, pp. 186–194. IEEE, 2017.

[42] D. Raja, D. Bowman, J. Lucas, and C. North. Exploring the benefits of immersion in abstract information visualization. In *Proc. Immersive*

*Projection Technology Workshop*, pp. 61–69, 2004.

[43] D. Ren, B. Lee, and T. Höllerer. Stardust: Accessible and transparent gpu support for information visualization rendering. In *Computer Graphics Forum*, vol. 36, pp. 179–188. Wiley Online Library, 2017.

[44] H. Rosling. Gapminder. *GapMinder Foundation http://www. gapminder. org*, 91, 2009.

[45] M. Saifee. Vr-viz. online: `https://www.npmjs.com/package/vr-viz`, 2018.

[46] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2017.

[47] R. Sicat, J. Li, J. Choi, M. Cordeil, W.-K. Jeong, B. Bach, and H. Pfister. Dxr: A toolkit for building immersive data visualizations. *IEEE transactions on visualization and computer graphics*, 2018.

[48] P. S. Strauss. Iris inventor, a 3d graphics toolkit. In *ACM SIGPLAN Notices*, vol. 28, pp. 192–200. ACM, 1993.

[49] P. S. Strauss and R. Carey. An object-oriented 3D graphics toolkit. In *ACM SIGGRAPH Computer Graphics*, vol. 26, pp. 341–349. ACM, 1992.

[50] R. M. Taylor, II, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser. Vrpn: A device-independent, network-transparent vr peripheral system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '01, pp. 55–61. ACM, New York, NY, USA, 2001. doi: 10.1145/505008.505019

[51] E. R. Tufte, N. H. Goeler, and R. Benson. *Envisioning information*, vol. 126. Graphics press Cheshire, CT, 1990.

[52] H. Wickham. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2010.

[53] L. Wilkinson. *The grammar of graphics*. Springer Science & Business Media, 2006.

[54] W. Willett, Y. Jansen, and P. Dragicevic. Embedded data representations. *IEEE transactions on visualization and computer graphics*, 23(1):461–470, 2017.

[55] Y. Yang, T. Dwyer, B. Jenny, K. Marriott, M. Cordeil, and H. Chen. Origin-destination flow maps in immersive environments. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018. doi: 10.1109/TVCG.2018.2865192

[56] Y. Yang, B. Jenny, T. Dwyer, K. Marriott, H. Chen, and M. Cordeil. Maps and globes in virtual reality. 2018.