VRIA - A Framework for Immersive Analytics on the Web

Peter W. S. Butcher
University of Chester
p.butcher@chester.ac.uk

Nigel W. John
University of Chester
nigel.john@chester.ac.uk

Panagiotis D. Ritsos
Bangor University
p.ritsos@bangor.ac.uk

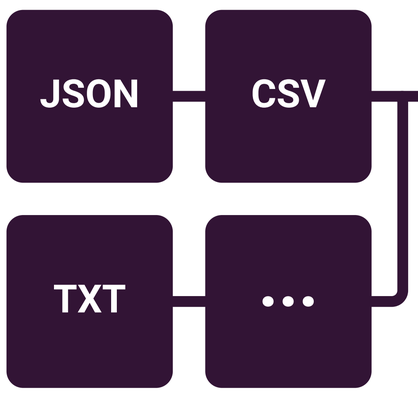
SCAN ME

github.com/vriajs

We present *<VRIA>*, a framework for building applications for Immersive Analytics using open-standard Web technologies. By combining WebVR, A-Frame, React and Redux, you can now write simple JSON configuration files to produce interactive and immersive Web-based visualizations that are ready to be experienced on every WebVR supported platform.

1 Dataset
Start with your data

Choose from one one of *<VRIA>*'s supported data types.



2 Configuration File
Based on Vega-Lite

<VRIA>'s configuration files are based on Vega-Lite specifications except now you can encode data in three dimensions.

```
// config.js

import dataset from './populations.csv';

export default {
  title: 'Populations over time (thousands)',
  data: { dataset },
  mark: { shape: 'box' },
  encoding: {
    x: {
      field: 'Year',
      type: 'ordinal',
      timeUnit: 'YYYY'
    },
    y: {
      field: 'Population',
      type: 'quantitative'
    },
    z: {
      field: 'Country',
      type: 'nominal'
    },
    color: {
      field: 'Country',
      type: 'nominal'
    }
  }
}
```

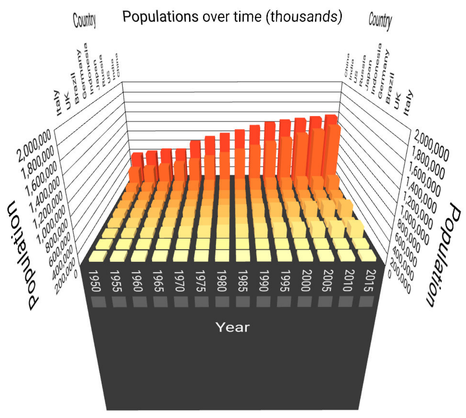
3 Application Code
React and A-Frame

<VRIA> can slot into practically any existing Web application, the only prerequisites are A-Frame and React.

```
// App.js

import React from 'react';
import ReactDOM from 'react-dom';
import * as AFRAME from 'aframe';
import { Scene } from 'aframe-react';
import VRIA from 'vria';
import config from './config';

class App extends React.Component {
  render() {
    return(
      <Scene>

        <VRIA config={config}/>

      </Scene>
    );
  }
};

ReactDOM.render(
  <App/>,
  document.getElementById('root')
);
```

4 WebVR Visualization
Interactive and Immersive

Experience your immersive data visualization in Virtual Reality on any device with a WebVR enabled browser.



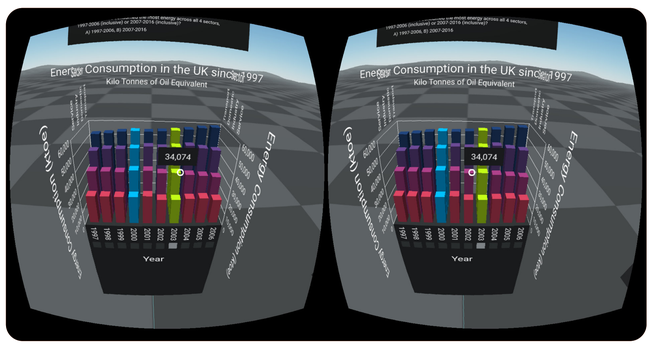
Built with WebVR, A-Frame, React and Redux

Interaction Components

Every *<VRIA>* visualization component has a corresponding set of interaction components which can be configured in the visualization configuration file. New interactions can be written with A-Frame and React and added to your application with *<VRIA>*'s API.

Interaction components work with a range of input devices including keyboard and mouse for desktop devices, gaze cursors for mobile devices, and controllers for 3DOF and 6DOF VR HMDs.

▼ Gaze cursor interaction with *<VRIA>* on mobile

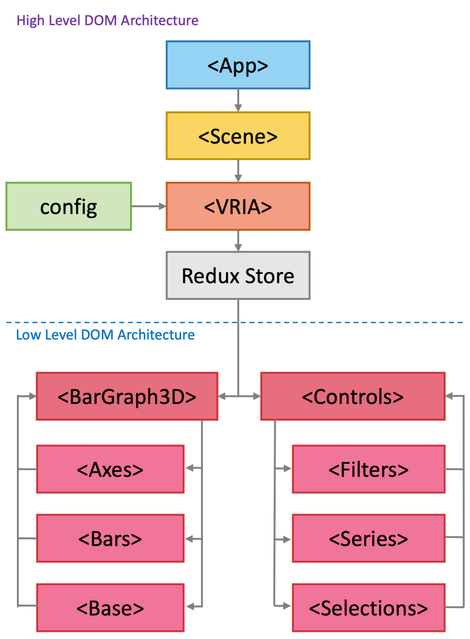


<VRIA> currently supports 3D bar charts (above) and multivariate scatter plots (right) with more abstract data visualization types planned.

Our framework is in ongoing development, with a number of enhancements planned. In the next phase we look to build more 3D vis components, along with corresponding interaction mechanisms. We also plan to integrate features that allow collaborative tasks in VR space.

Architectural overview of an example application built with *<VRIA>*

The exact structure and implementation of the high-level DOM architecture of an application that makes use of <VRIA> is up to the user, and there is no requirement for the whole application to be written in React. <VRIA> can be integrated into existing applications, with the only requirement being that the overarching application makes use of A-Frame scenes. The resulting low level DOM architecture is generated from the JSON config file that is passed to the <VRIA> React component.

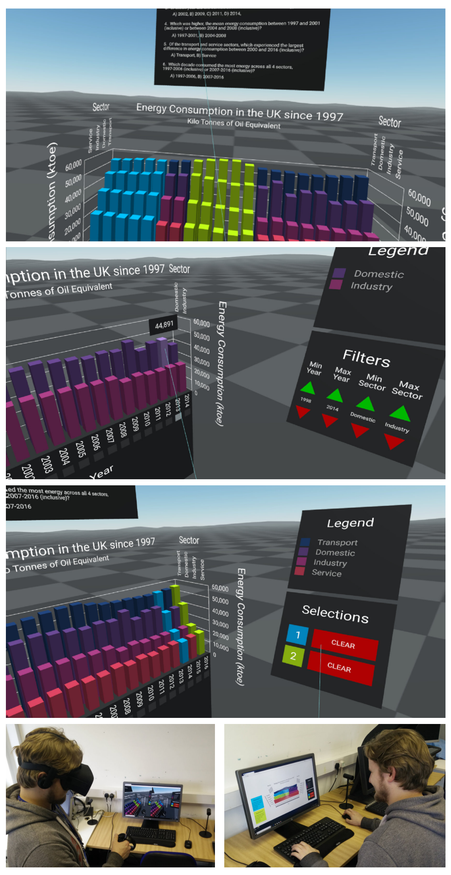


- **Top Level React Node:** React component containing an A-Frame scene component and any other application code.
- **A-Frame Scene Component:** Contains other A-Frame components and the *<VRIA>* React component.
- **Visualization Config File:** A JSON file containing a description of the visualization and controls.
- ***<VRIA>* React Component:** Placed within the A-Frame scene, this component is passed a JSON visualization config.
- **Redux Store:** Contains the application state. It is connected to the visualization and control components.
- **Visualization and Control Components:** These components map data to other A-Frame React components.

User experience evaluation of a <VRIA> use case

We conducted a user study to evaluate a use-case scenario, produced with the current version of <VRIA>, in terms of usability, user experience and the presence of any simulator sickness symptoms. By evaluating our framework at this intermediate stage we aimed to inform design decisions going forward, both in terms of designing <VRIA>'s API, as well as providing guidelines to the users, for creating more usable visualizations. We collected data using the Simulator Sickness Questionnaire (SSQ) methodology, the System Usability Scale survey, timings of task completion and post-experiment open-ended questions. An overview of our findings are presented below. See our paper for detailed findings.

▼ Experiment tasks in action



▲ Box plot of SUS scores. The usability of the 2D desktop set-up is high, compared to the pseudo-3D variant, with the HMD-based system close in perceived usability to the 2D set-up.

▲ Box plots showing the difference in completion times across the three test conditions.

P. W. Butcher, N. W. John, and P. D. Ritsos. VRIA - A Framework for Immersive Analytics on the Web. In CHI'19 Extended Abstracts, May 4-9, 2019, Glasgow, Scotland UK, 2019. doi: 10.1145/3290607.3312798