PRIFYSGOL
BANGOR
UNIVERSITY

School of Computer Science
College of Physical & Applied Sciences

Parallel Coordinate Plots in Immersive Analytics

Joshua Henderson

Submitted in partial satisfaction of the requirements for the
Degree of Bachelor of Science
in Computer Science

Supervisor Dr. P. Ritsos

May 2018

Acknowledgements

> *A display connected to a digital computer gives us a chance to gain familiarity with concepts not realisable in the physical world. It is a looking glass into a mathematical wonderland.*
>
> — **Ivan E. Sutherland, widely regarded as the ’father of computer graphics’.**

I would like to thank my supervisor, Dr. P Ritsos, for his endless guidance and enthusiasm in the development of this project. Its taken a lot of stress and time, but I’m happy to call it complete. I would also like to thank my family for their constant, if confused, support, and all of the questionnaire participants who helped test the final implementation.

Statement of Originality

The work presented in this thesis/dissertation is entirely from the studies of the individual student, except where otherwise stated. Where derivations are presented and the origin of the work is either wholly or in part from other sources, then full reference is given to the original author. This work has not been presented previously for any degree, nor is it at present under consideration by any other degree awarding body.

Student:

Joshua Henderson

Statement of Availability

I hereby acknowledge the availability of any part of this thesis/dissertation for viewing, photocopying or incorporation into future studies, providing that full reference is given to the origins of any information contained herein. I further give permission for a copy of this work to be deposited with the Bangor University Institutional Digital Repository, and/or in any other repository authorised for use by Bangor University and where necessary have gained the required permissions for the use of third party material. I acknowledge that Bangor University may make the title and a summary of this thesis/dissertation freely available.

Student:

Joshua Henderson

Abstract

Data visualisation is a vast industry, requiring many bespoke tools and technologies. By applying the principles of the relatively new and burgeoning area of web-based virtual reality, we can create cutting edge immersive visualisations to display various types of data. In this project, we explore the feasibility of a web-based virtual reality parallel coordinate plot to display high-dimensional multivariate data in way that is as immersive for the user as possible.

The end result is a solution that makes use of the A-Frame and D3.js APIs, alongside the HTC Vive head-mounted display and controllers to create an immersive data experience.

Contents

List of Figures

Chapter 1

Introduction

1.1 Aims and Overview

This project intends to create an immersive, virtual environment for the dynamic visualisation of high-dimensional multivariate data. In order to achieve this, I will create a web application to generate an interactive parallel coordinate plot using the A-Frame virtual reality framework. This is will then allow it to be displayed platform independent through any current-generation web browser that has WebVR capability, on an Oculus Rift, HTC Vive, or other compatible head-mounted display.

The web application will allow customisation and movement of all axes, with dynamic order detection and line regeneration. This, and all other interactivity, will be achieved using the virtual reality hardware's controller system, such as the Oculus Touch platform or the Vive Controllers. A small menu for manually editing settings or displaying individual datum will be attached to the in-world representation of the controllers.

In brief, the objectives for this project are:

1. Explore the use of Mixed Reality.

2. Convert a CSV file of multivariate data into an A-Frame representation.

    - Start with pregenerated car properties data set.

- Expand to cover data sets of any dimension.

3. Implement a way to individually view each datum.

4. Implement a way to customise the look and position of each axes.

1.2 Motivation

Immersive Analytics is a relatively new initiative that combines the disciplines of data science, mixed reality, and 3D interfaces to explore and develop new interaction and visualisation technologies to expand the realm of data analytics and support further analytical reasoning and decision techniques.

With the advance of technology and the increasing popularity of data warehouses, we have the ability to create, utilise, and visualise larger sets of data than previously capable. Freely available data sets from various sources such as US [15], UK [14], or European Union [13] governments, including census data, military spending, or even environmental reports; the World Health Organisation; or even tech industry giants such as Google, Amazon, or Facebook are easily obtainable with minimal searching required. Such large sets of high-dimensional multivariate data presents a problem in that traditional methods of visualisation can be easily overwhelmed as the size of the data set increases, resulting in the need for new applications and techniques that take advantage of the latest hardware available. This project explores the feasibility of integrating an immersive, virtual environment with a parallel coordinate plot using head-mounted display devices.

1.3 Thesis Statement

We should explore the use of burgeoning technologies to augment the visualisation and interaction of data sets as the ever-expanding access to data of the information age increasingly antiquates current work.

1.4 Dissertation Structure

The structure of this dissertation is as follows:

Chapter 2 examines background work in the areas of Augmented and Virtual Reality, and it's in use in consumer, military, and medical applications.

Chapter 3 details the process of designing, implementing, and testing the A-Frame Parallel Plots project.

Chapter 4 compares the completed work to the original aims, objectives, and research questions to determine if the project is suitable for it's purpose.

Chapter 5 examines the wider impact of this work, including professional, social, technological, legal, and ethical implications where ever there are any.

Chapter 6 takes a look at the findings of this project, and the lessons that have been learned. It will determine where the project can be developed further in the future.

Chapter A provides an attached copy of the code listings and any other resources used to complete this project.

Chapter B provides an attached copy of the poster presented at the dissertation expo.

Chapter 2

Background

2.1 Mixed Reality

Mixed Reality is an idea that has existed since the 1960s. It is the area in between on a scale from the completely real to the completely virtual, also known as Milgram's continuum or the virtuality continuum [23]. This means that mixed reality ranges from the virtual-enhanced real world of augmented reality, where computer-generated perceptual information including visual, auditory, or haptic feedback is overlaid onto a view of the real world, to the real world-enhanced virtual world of augmented virtuality, where physical elements are dynamically integrated into and can interact with a predominantly virtual space (see Fid. 2.2).

Figure 2.1: Reality-virtuality continuum.

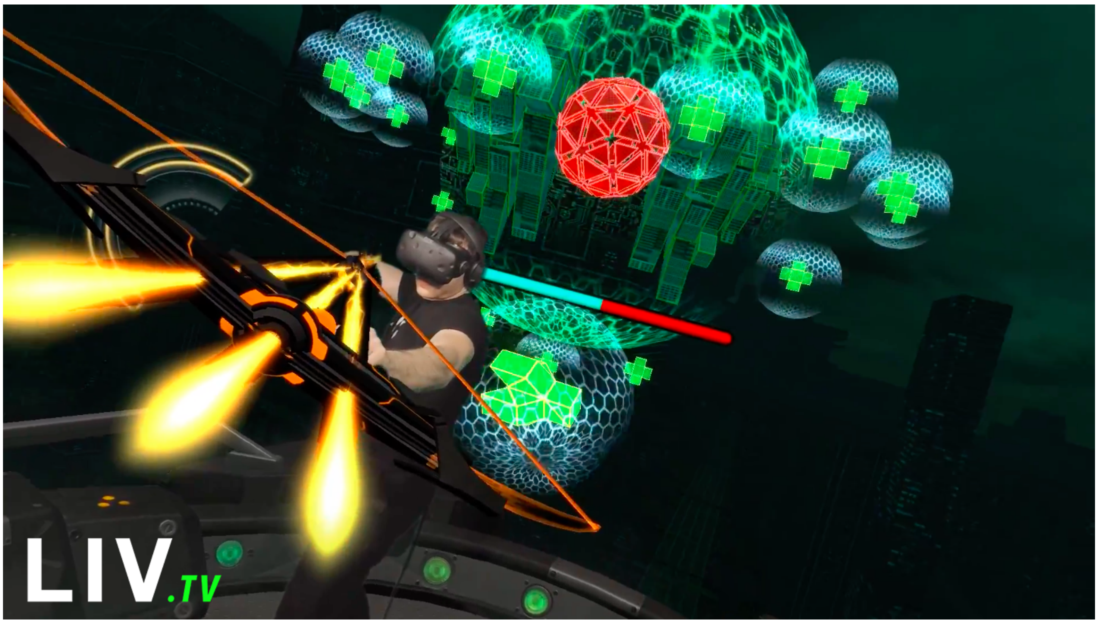


Figure 2.2: Augmented Virtuality, image by Pierre-faure.

2.2 Immersion

2.2.1 Virtual Reality

The two main contenders for the title of first virtual reality head-mounted display system are often thought of as the 'Sword of Damocles' and the Philco HMD.

The Sword of Damocles [26] was created in 1968 by computer science pioneer Ivan Sutherland in his quest for what he termed 'the ultimate display' [25]. Working with the limited resources available back then, the device was only capable of displaying simple wire-frame rooms in a stereoscopic display. To enable the changes in perspective necessary based on the position of the user's gaze, a rudimentary form of head tracking was developed. The device was attached to a mechanical arm suspended from the ceiling, inspiring the name, and fastened to the user's head (see Fid. 2.3).

The system was further developed, and by the end of the decade its previously separate components were fully integrated into a single fully functioning HMD

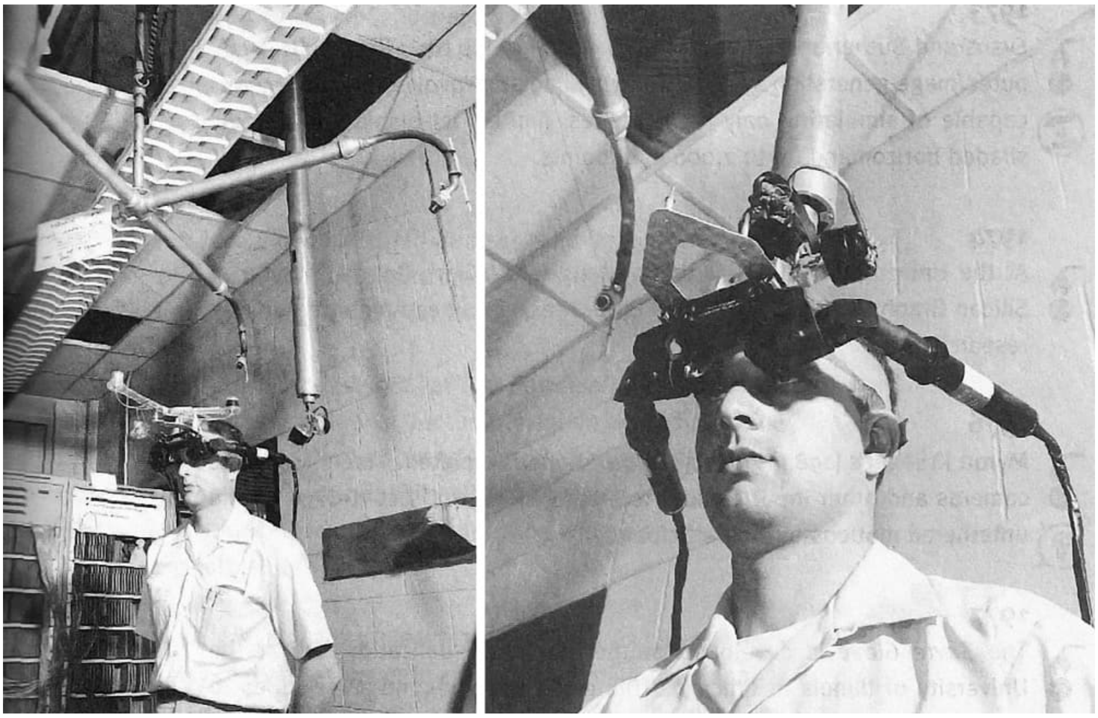


Figure 2.3: The Sword of Damocles

system. The unit was also partially translucent, resulting in it often being cited as a precursor to augmented reality technology too.

The Philco HMD, known as Headsight, took a different approach. Developed in 1961, the device used a magnetic motion tracking system linked to a closed circuit camera. Its purpose was a military application, allowing the remote viewing of areas and situations where it was deemed too dangerous for a human viewer to physically visit. The device created a sense of telepresence [20], as the tracked head movements controlled the position of the remote camera.

The term 'Virtual Reality' was possibly first coined by French playwright Antonin Artaud in his 1938 book 'The Theater and Its Double' which speaks of the theater as 'la réalite virtuelle' [2], a reality both illusory and fictitious; though it was arguable popularised by founder of VPL Research Jaron Lanier. VPL Research, founded in 1984 [4], stood for 'Virtual Programming Languages' and was one of the first companies to develop and sell virtual reality products, including the Data Glove (see Fig. 2.4).

Figure 2.4: The EyePhone and the Data Glove, products of VPL Research

The idea of data gloves dates back to 1977 with the Sayre Glove, created by the Electronic Visualisation Laboratory at the University of Illinois. The device consisted of light based sensors and flexible tubes with a light source at one end and a photocell at the other. When the fingers bend, the amount of light hitting the photocell varies [24]. VPL Research took this idea further, developing the Data Glove in 1987. Using 6502-type micro-controllers, the glove monitored 10 finger joints and allowed six degrees of freedom of the hand’s position and orientation [33]. It operated in real time and did not rely on line-of-sight resulting in a clear improvement over existing camera-based techniques. The company also developed other technologies, such as the EyePhone, an early HMD using Fresnel lenses [7], featured in the 1992 science-fiction action-horror ’The Lawnmower Man’, supposedly an adaptation of the Stephen King short story of the same name (though this was debated in a lawsuit by Stephen King himself); and the Data Suit, a full-body outfit with sensor to measure arm, leg, and trunk movement.

In 1985, NASA’s Ames Research Center created the Virtual Interface Environment Workstation, or VIEW. With the availability of small, cheap LCD televisions, alongside a range of more powerful computer graphics machines from Silicon Graphics Inc, NASA was able to create it's first prototype HMD. The first iteration of the VIEW, known then as the Virtual Environment Display, was assembled from a motorcycle helmet, wide angle stereoscopic optics, a Polhemus magnetic head-position tracker, and Watchman LCD displays. The device was innovative in that it incorporated the first data-glove into virtual reality. This glove was able to measure finger movement, plus hand position and orientation, allowing the user to interact with the virtual world. Users could grasp objects and perform a variety of hand gesture commands, such as pointing the index finger to fly through the environment. These gestures were conceived and implemented by Warren Robinett [10].

Virtual reality gaming was only an idea until the early 1990s, when Japanese video game developer SEGA start development on the SEGA VR unit, an unreleased add-on for the SEGA Mega Drive game console. Based on the IDEO HMD, SEGA VR made use of LCD screens in its visor, stereo headphones, and inertial sensors [18] to track and react to the user's head movements. Designed to be lightweight as to not hurt with prolonged wear, the HMD could track head movements at a rate of 100Hz, which combined with it's stereoscopic 3D technology could immerse the user in a virtual world. Set to launch in December 1993, the project was quietly cancelled around 1994/1995 due to increasing costs, and concerns of eye damage, headaches, and motion sickness associated with early HMD technology. SEGA themselves claimed at the time that the project was stopped as, in their words, the VR was so 'real', that users would attempt to move and injure themselves [18].

Supported by enthusiasts on the crowd-funding platform Kickstarter, Palmer Luckey founded OculusVR in 2012. The first development kit for the Oculus Rift used a 7-inch LCD screen and initially only had a 125Hz head-tracker, thought it was increased to 250Hz with firmware. Later versions of the development kit had the tracker replaced with a new 1000Hz solution produced by Oculus themselves, combining three-axis gyros, accelerometers, and magnetometers. The kit was released in March 2013 [31]. The second development kit, shipped in July 2014, improved over the previous design with a higher-resolution low-persistence OLED display, with a higher refresh rate, new positional tracking, a detachable cable, and the omission of an external control box.The display itself was revealed to incorporate a modified Samsung Galaxy Note 3 smart-phone display [29]. Finally, in March 2016, Oculus began shipping its latest consumer version of the Oculus Rift. Sporting per-eye displays running at 90Hz, with a higher combined resolution than the previous development kit 2, the device was refined for consumer use, focusing on ergonomics and aesthetics [30]. The consumer version features tracking LEDs on the back of the headset, allowing full 360-degree tracking, alongside it's room tracking sensors, known as Constellation [11]. A controller system known as Oculus Touch as also developed for release.

Starting in 2012 with a system made up of a simple HMD, a camera, and a few AprilTags [9] (larger, simpler QR codes), Valve started the research and development of gaming applications with a focus on freedom of movement and 'room-scale' VR. Initially Valve didn't intend to build their own headset, preferring to collaborate with OculusVR on it's tracking to, as they put it, 'drive PC VR forward'. However, at some point over the course of 2014, their working relationship deteriorated, resulting in HTC and Valve meeting in late spring [9]. Valve's room-scale VR started off using a system of AprilTags placed around the area to provide tracking, but this was deemed too impractical. Other solutions were explored, such as dot tracking and laser tracking. Dot tracking consisted of covering the headset and controllers with small dots, usually with LEDs. A stationary camera then uses machine vision to determine the position of these dots to provide tracking, this is how the Oculus Rift's Constellation system operates. Valve, however, settled on a different plan. Laser tracking offered a higher accuracy, but was much more difficult to implement on a room scale. In this approach, base stations around the room house emitters that rotate rapidly to send a laser across the room, alongside LEDs that flash at 60Hz [9]. The headset must detect these inputs, determining its position and orientation based on the gaps in time between the laser and LEDs hitting each sensor. This proved to be a much more complicated, but more effective solution. In October 2014, the first prototype was shown to select developers, requiring them to sign an non-disclosure agreement just to look at the actual NDA. In January 2015, the pre-developer edition of the HTC Vive was debuted, featuring a front-facing camera (part of the Chaperone system designed prevent harm by displaying real-world images on top of the virtual world) and an in-built microphone to answer calls while inside the headset. In April 2016, the consumer edition was released [28].

2.2.2 Augmented Reality

Mixed Reality isn't just about head-mounted displays however. In 1974, Myron Krueger, a computer artist known for developing early interactive works, established a new artificial reality laboratory that he titled the 'VIDEOPLACE'. The idea was to avoid the encumbrance of goggles and gloves using various technologies such as video cameras, projectors, special purpose hardware, and onscreen silhouettes of the users, placing them within an interactive environment. Movements recorded on video were analysed and transferred to the silhouette representations to allow the users to see their actions and also interact with each other. This gave them a sense of presence regardless of the lack of tactile feedback, strong enough that users even pulled away whenever their silhouettes intersected with others [20].

Inspired by Plato's 'The Allegory of the Cave', an exploration of the concepts of reality and human perception, Thomas Defanti and Dan Sandin conceived the idea of the CAVE Automatic Virtual Environment in 1991. For his exploration, Plato used an analogy of a imprisoned man who's entire reality was defined by shadows projected on the wall of the cave he is chained to, interpreting that our perception of reality is but a construct created by our minds and the human condition is forever bound to impressions received through the senses [8]. The CAVE is a cube-shaped room where the walls, floors, and ceilings are projection screens (see Fig. 2.5). The user wears either a HMD, a heads-up display or simply a set of 3D glasses, and can interact with the environment using devices such as wands, joysticks, or data-gloves. Objects appear floating in the air, allowing users to walk around them and perceive them as they would in reality. This was initially done with electronic magnetic sensors, adding the limitation that non-magnetic materials must be used in the CAVEs construction, though it was eventually converted to infrared cameras to get around this. Other sensors in the user's headgear help to provide movement tracking, and allow the video to constantly adjust itself to retain the user's perspective. Speakers provide 3D sound to further complement the immersion. The first iteration of the CAVE was debuted at the 1992 SIGGRAPH conference, and a second iteration was developed in 1994.

Developed by the semi-secret research and development arm of Alphabet, Google X [12], Google Glass is an optical head-mounted display shaped like a pair of eyeglasses, released as a prototype in May 2014. The device is controlled using natural language commands, initiated with the phrase 'OK Glass', alongside a small touchpad on it's side to allow swiping gestures. It uses 'liquid crystal on silicon' technology [16], with various beam splitters to reflect the displayed image onto the eye. The programs it could run include various applications created by both Google and third-party developers, including Google Maps, providing active directions as a user travelled; informative apps, such as Gmail, Evernote, and news providers; and social network sharing capabilities, for Facebook or Twitter. The device included a 720p camera, allowing users to record video and resulting in various criticism and legislative action over concerns about privacy and safety [21]. In January 2015, the device was discontinued, to be later re-released with the Google Glass Enterprise Edition in July 2017.

Initially known as Project Baraboo [17], the Microsoft HoloLens is a head-mounted pair of smart-glasses running an integrated Windows 10 computer with a custom-made Microsoft Holographic Processing Unit (HPU) [22] to support its CPU and GPU. Featuring a HD 3D optical display, spatial sound projection, and movement, gesture, and voice sensors, the HoloLens is designed to be an immersive holographic experience that is light point and radiant rich. Its principal AR ability is the capability to perform a real-time scan of the environment to create a mesh of an XYZ coordinate plane **Spatial mapping**, allowing realistic 3D projections to be anchored onto real world objects within the mesh. Alongside gaze, gesture, and voice commands, interactivity is expanded with a hand held clicker with a single button and rotational tracking.

Figure 2.5: CAVE Automatic Virtual Environment

2.3 Visualisation

The Advanced Technologies Group within the peer-to-peer ride-sharing company, Uber, worked with it's Data Visualisation team [3] to improve how their self-driving vehicles perceive and interact with the world [5]. To do this they created a new web-based application using the 'react-map-gl' and 'deck.gl' APIs that collates various data sources from online and offline testing, such as ground surface scans, lane boundaries, other preprocessed mapping information, and run-time vehicle logs, to simulate an autonomous vehicles journey through an area and display it's sensory information, how it predicts objects to move, and what it plans to do in reaction. This visualisation platform is heavily customisable, allowing each team withing the Advanced Technologies Group to modify and tweak it to their individual needs.

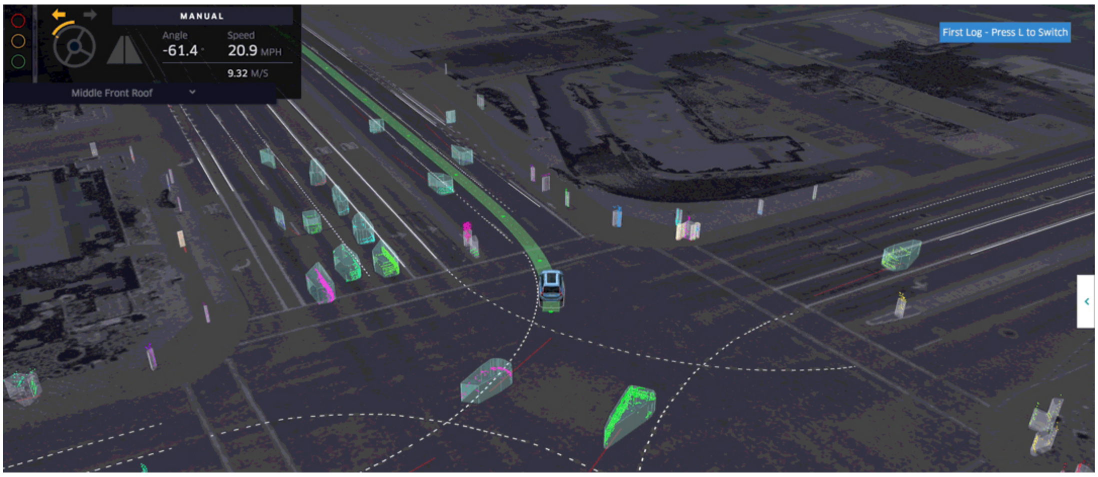


Figure 2.6: Uber's Self-Driving Car Visualisation Platform

2.3.1 Engagement

The success of a visualisation is determined by many factors, the most important of which being performance, usability, memorability, fun, and engagement.

For this project, we'll focus on measuring user engagement, using VisEngage [19], a self-assessment questionnaire that is designed to provide insight into 11 different characteristics of user engagement. In their study, Hung and Parsons searched through various literature from related domains to identify engagement characteristics, merging terms that were too broad or vague. They identified 57 in total, reducing that to 11 terms that had the highest frequency and relevance in InfoVis. These characteristics were: Aesthetics, Captivation, Challenge, Control, Discovery, Exploration, Creativity, Attention, Interest, Novelty, and Autotelism. They then identified two each items for each characteristic that were more concrete than their abstract sources, and therefore usable in questions with a seven-point Likert scale from strongly disagree (1) to strongly (7). These answers are totalled to provide an engagement score for use in evaluation.

2.4 Relevant Technologies

2.4.1 WebGL

Integrated into all current web standards for browsers, WebGL is a JavaScript API that allows GPU-accelerated use of physics, image processing, and effects as part of the web page canvas to render interactive 2D and 3D graphics in a web browser without the use of external plug-ins [27]. WebGL programs are made up of JavaScript control code and shader code written in the OpenGL Shading Language (GLSL).

WebGL is designed and maintained by the non-profit Kronos Group.

2.4.2 WebVR

Initially conceived in 2014 by Vladimir Vukićević from Mozilla, WebVR is a currently-experimental JavaScript API providing web browser-based support for virtual reality devices such as the HTC Vive, Oculus Rift, Google Cardboard, or OSVR. Various members of the Mozilla team and the Google Chrome team contributed to the version 1.0 release in March 2016, with members of Microsoft joining to collaborated on version 2.0.

Currently versioned 1.1, the API was designed with the following goals [32]:

- Detect available virtual reality devices.
- Query the devices capabilities.
- Poll the device's position and orientation.
- Display imagery on the device at the appropriate frame-rate.

The API uses WebGL, with necessary camera settings and device interactions, to expose new interfaces such as VR Display and VR Pose.

2.4.3 A-Frame

Developed within the Mozilla VR team in 2015, A-Frame [1] was created to allow developers and designers to create VR experiences without knowing WebGL, entirely through HTML. The framework is an open-source project, now maintained by Mozilla and members of the WebVR community.

All 'under-the-hood' work, such as canvas rendering, render loops, lighting, and setup, are performed by A-Frame itself, initialised with 1 HTML line ('<a-scene>'). The framework runs on an entity-component architecture using Three.js meaning that every object in a scene is an entity, whereas components, written in JavaScript, are composable and reuseable modules that can be linked to entities to provide functionality, appearance, or behaviour.

The framework also includes a visual inspector tool that can be enabled in the browsers from any live A-Frame scene, allowing for ease of development.

2.4.4 D3.js

Successor to earlier data visualisation framework Protovis, D3.js [6], released in 2011, was designed to be a more expressive framework that also focuses on web standards and improved performance. D3, standing for Data-Driven Documents, makes uses of SVG, HTML5, and CSS standards to provide a data visualisation framework with greater control over the final visual result, useful for creating interactive web graphics, information dashboards, and showing GIS map making data. Owing to the exportable nature of SVG, D3 graphics can also be used in print publications.

Using pre-built JavaScript functions, D3 allows developers to select elements, create SVG objects, style them, and add transitions, dynamic effects, or tool-tips. Styling can also be provided through CSS. Data sets in various formats can be bound to SVG objects, allowing the dynamic generation of graphic charts and diagrams.

The D3.js API currently contains several hundred functions, grouped into the following units: Selections, Transitions, Arrays, Math, Colour, Scales SVG, Time, Layouts, Geography, Geometry, and Behaviours.

Chapter 3

Body Chapter(s)

3.1 Design

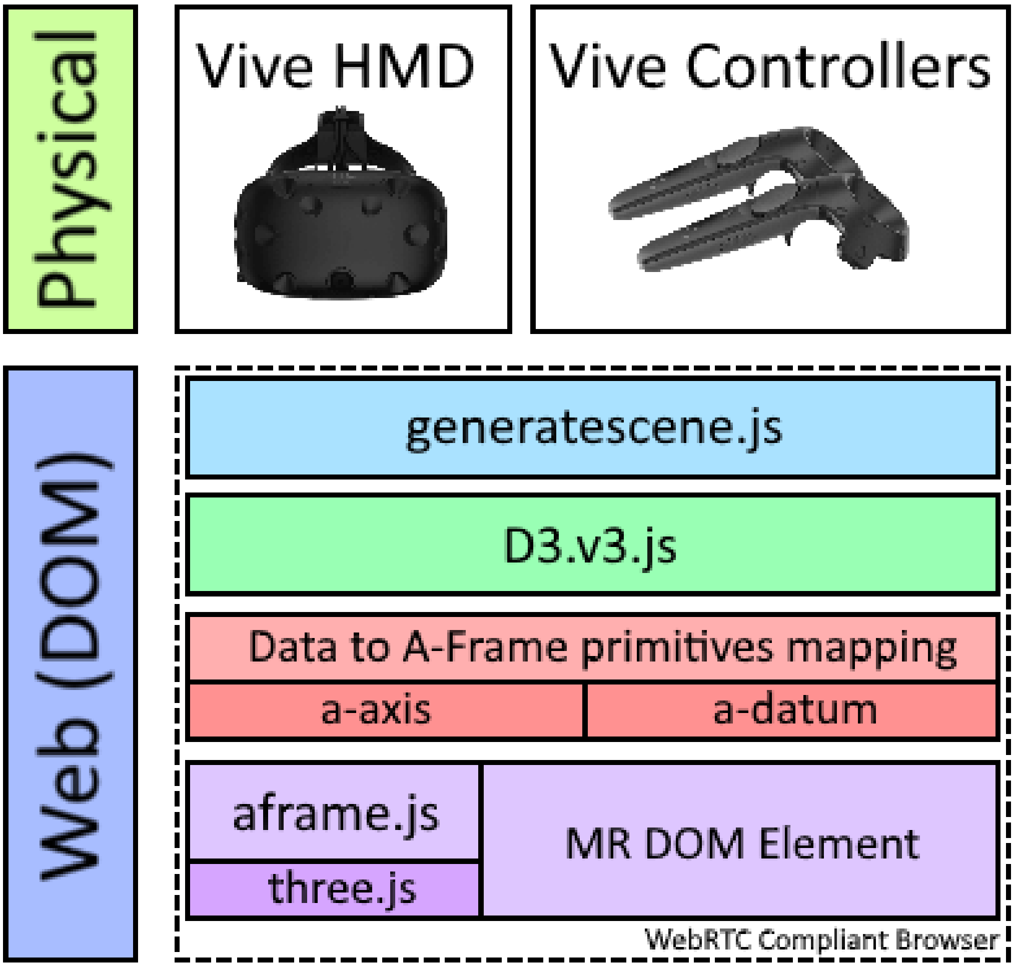


Figure 3.1: Anatomy of the project, featuring the various APIs used.

3.1.1 Interaction

There are multiple ways that controller interaction can be handled, each with their own pros and cons.

Idea 1 A line is raycast from the controller. Upon use of the trigger button, any axis that is intersecting the line is selected. The axis' position is then tethered to the end of the raycast line, moving to wherever it intersects with the floor plane. Upon use of the trigger again, the position is final and the axis is untethered.

This methods provides the most obvious feedback to the user, making it much easier to use and understand.

However, this method provides the most overhead in computation. Each slight move of the controller would results in the axis calculating its order in the list, and constantly regenerating its surrounding sectors. Since sector generation is the most resource intensive part of the visualisation, this would provide serious latency issues that may make the program unusable.

Idea 2 A line is raycast from the controller. Upon use of the trigger button, any axis that is intersecting the line is selected, and changes colour. A new phantom axis is created, with zero data connections, and tethered to the end of the raycast line, moving to wherever it intersects with the floor plane. Upon use of the trigger again, the selected axis is moved to the position of the phantom, and the phantom axis is deleted.

This method solves the latency issue. Since the selected axis only has its position changed once, its order is only calculated once, and its surrounding sectors are only regenerated once.

However, this method is less obvious to user, and may result in some confusion, such as them thinking they may be creating new axes, rather than moving the selected one. It also provides the most overhead in implementation, owing to it being a much more complex method.

3.2 Implementation

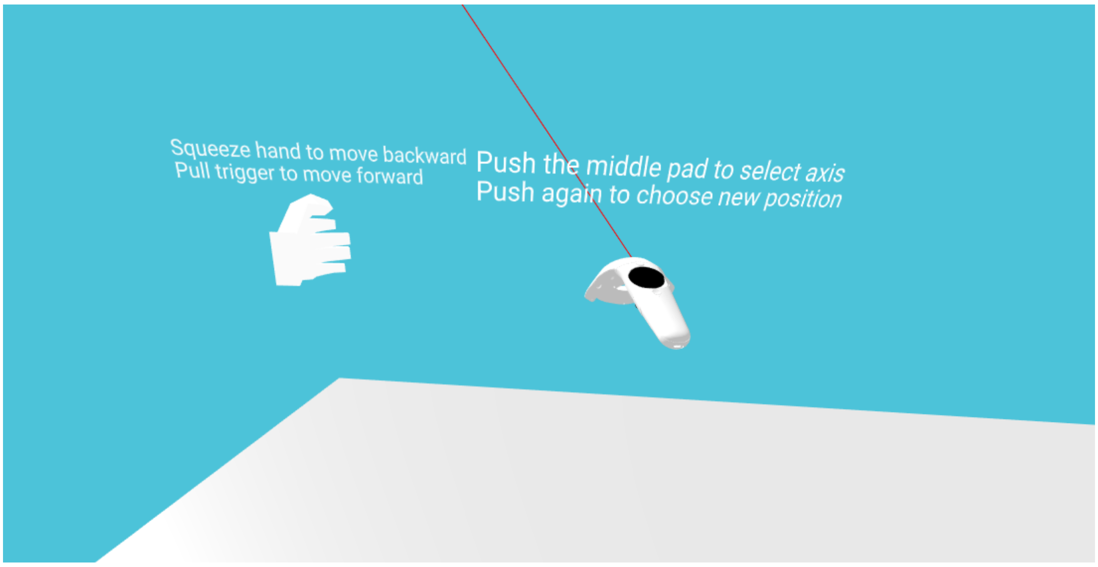


Figure 3.2: Models representing the Vive Controllers when in VR.

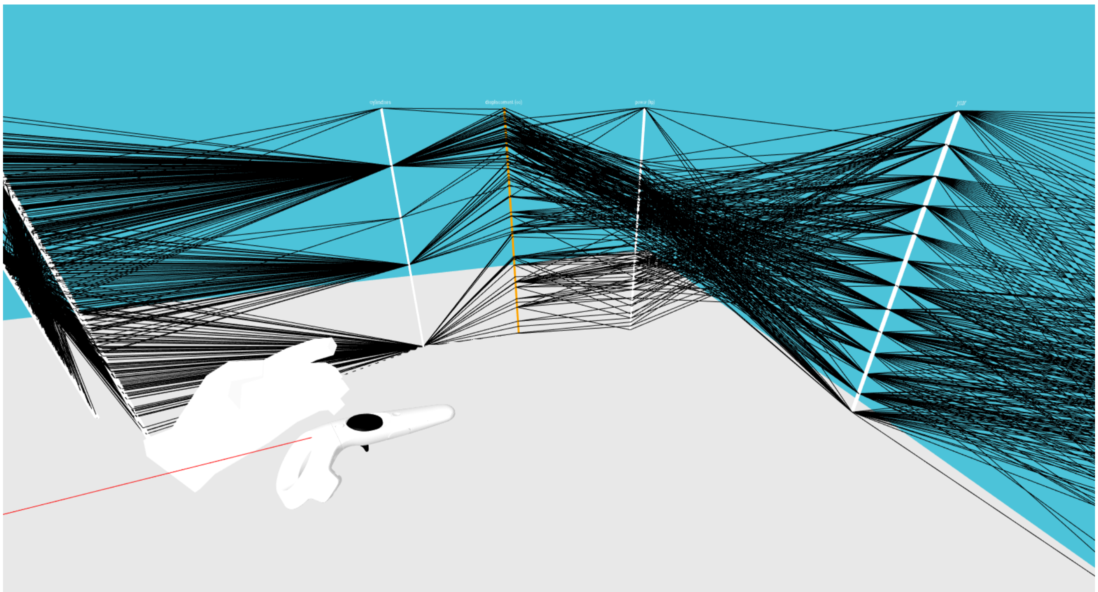


Figure 3.3: An axis is selected, and displayed in orange.

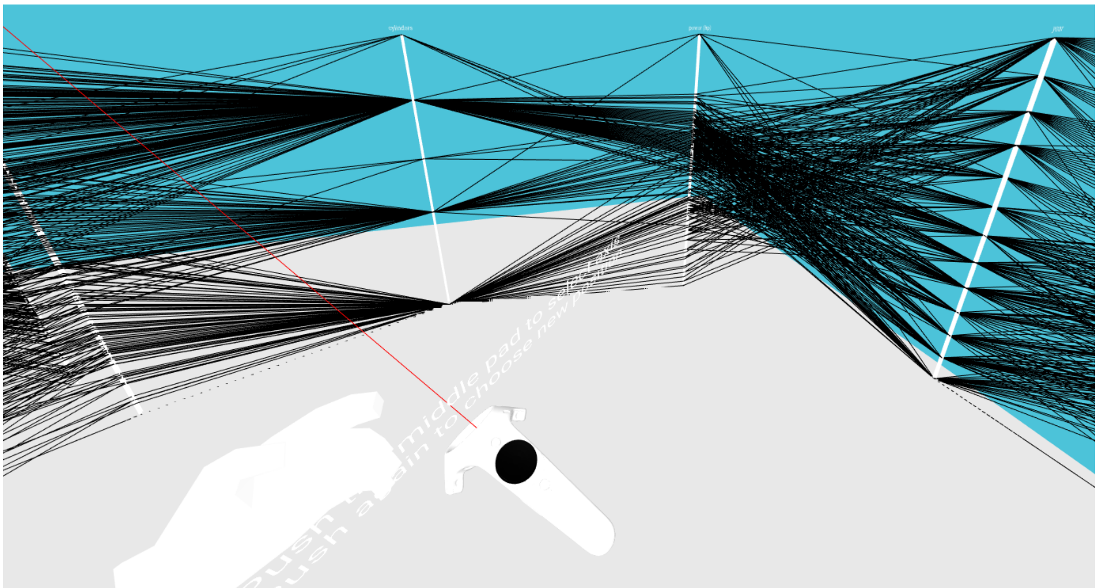


Figure 3.4: The selected axis has been moved.

3.2.1 Main Process

Overview

To produce an immersive parallel coordinate plot from a set of CSV data, the program goes through three main stages:

1. Read in the CSV file, committing it to memory, and generate scales.
2. Strip the name column at position zero, and generate an axis for each of the remaining columns
3. Generate the connections between each axis.

Data

The program uses the D3 CSV function to parse the provided file into an array. This parsed array is then mapped to numeric values and stored in a global variable for future use.

Separate the column titles into a new global variables, using the D3 filter function to remove the name column, in the case of the sample data, this would be the car model. Create a new linear scale ranging from zero to the maximum width, with a domain from zero to the number of column names. This will represent the X position, allowing equal spacing between each axis at generation.

Axes

Next the program must create an 'a-axis' entity to represent each numerical axis, meaning all but the name column. To do this, it uses D3 to select all 'a-axis' entities within the 'a-scene' element of the document object model. At the time of initialisation, no 'a-axis' entities actually exist yet, but this allows us to use the D3 select function to easily append new entities.

The data function of the D3 selection then allows us to join the array of data to the current selection, to quickly create new axis entities with the append function.

Finally, the each function of the D3 selection allows us to modify the properties of the axis using its provided datum. The column name is attached using the datum; the position order set with the datum's index; position coordinates set using an offset for Z and passing the index to the previously created scale for X; height set using a default value; and the minimum and maximum values for the column are retrieved using D3 extent and also stored in the axis entity.

An 'a-text' element is appended as a child of the axis, positioned just above it and given the axis' name as its value.

Datum

With the array stored and the axes created, the program now needs to generate the individual connections between each axes, using the data within the array. It does this sector by sector, each sector representing all of the connections between two adjacent axes.

First, validation is performed, checking if there are axes to both the left and right of the sector, to ensure the sector actually exists. Then, two scales are generated to represent the Y coordinate of the connection on each of the axes. The range of the scales is from its axes' height properties to zero, while the domain is from its axes' minimum value property to its maximum value property. Then the program attempts to clear all existing connections within the sector, in the event that it is being regenerated.

Then a new datum element is appended and its sector is set. Each connection between the axes is represented by a single line component, attached to the sector's datum element, with only three properties: the first 'XYZ' coordinate, the second 'XYZ' coordinate, and the colour to display it with. The X and Z positions of the first coordinate are retrieved from the respective positions of the left axis, and the Y position is generated by indexing the data record with the left axis' column name to retrieve its column value, and then passing that to the left scale to retrieve its position in world space. The same is done for the second coordinate using the positions and scale of the right axis.

The two coordinates and colour of the connection is set, and the process is repeated for every record in the array.

3.2.2 Components

A-Frame uses an entity-component architecture, entities being container objects acting as the base of all objects, and components being reusable modules that are attached to entities to provide its appearance, behaviour or functionality.

In this program, entities are registered with mappings and default components to allow their quick deployment, with the relevant component automatically attached. Mappings are created to link HTML attributes to component properties of the same name, enabling the change of properties through D3's 'attr' function.

In this project there are two main components:

- Axis: represents a single axis, attached to 'A-Axis' entities.
- Datum: represents a single connection between two axes, attached to 'A-Datum' entities.

Each component definition will consist of the component's schema, or it's properties, and it's update function, which triggers whenever it detects a property change. Each component also contains 'init' and 'remove' functions, which are the same for both components as they are simply as follows:

- Init: Creates either a new 'THREE.LineBasicMaterial' or 'THREE.MeshBasicMaterial' using the values from the relevant properties. Buffers geometry with relevant attributes for position and scale. Creates new 'THREE.Line' or 'THREE.Mesh' geometry with the correct position, material, and attaches it to the component.
- Remove: Removes the previously generated geometry from the component.

Axis

Schema

Property	Type	Default
colour	colour	'#FFF'
name	string	' '
positionOrder	int	1
positionX	number	1
positionZ	number	1
height	number	500
maxValue	number	500
minValue	number	0
thickness	int	1

Update On detection of a change in its world coordinates, the axis needs to update its geometry. However, first it does a few checks on its position in the axes order.

If its new X position if lower than the X position of the axis before it in the list (if it exists), it gives its current order number to the previous axis, and then decrements its own order. The same check works in reverse if its X position is above the X position of the next axis in the list.

It then needs to trigger its data connections to regenerate. Rather than regenerate all connections in the plot, which would cause significant latency, the sectors (a sector being all the connections between the current axes and the one next in the list) it regenerates are dependent on its order in the list. The first axis only regenerates its own sector, and the next one. The last axis only regenerates the sectors 1 and 2 positions before it, as the final axis does not have a sector of its own. All other axes regenerate 5 sectors, centred on itself, with checks to ensure its within index bounds.

Datum

Schema

Property	Type	Default
colour	colour	’#FFF’
sector	int	0

Update Since most of the main calculation is performed by the axis, each data connection only needs detect changes in its coordinates and material, and change its geometry accordingly.

Laser Listener

The laser listener, or ’controller-listener’ as it is referenced in the code, controls the movement of the axis. Applied to the right controller alongside the raycast-firing ’laser-controls’ component, the laser listener applies an event listener that triggers whenever whenever both the middle button is pressed and the raycast intersects with an element.

If there is not already an axis selected, this listener detects whether the intersected element has an axis component, and if so records a reference to the axis and its original colour, then sets its colour to orange as an indicator that it has been selected. If there is an axis selected when the event emits, then the axis’ X and Z coordinates are replaced with the coordinates of the intersection point, its colour is reset to the original colour, and the reference is discarded.

Movement Listener

The movement listener, or ’movement-listener’ as it is referenced in the code, controls the movement of the camera view. It is applied to the left controller, alongside a ’hand-controls’ component that wraps other controls components and emits the necessary events. Pressing the grip button(s) moves backwards, while pressing the trigger button moves forward.

Init The initialising function serves to add variables representing the button states to the element, and binds the relevant functions as event listeners. Each button has an down and up listener, setting its relevant Boolean to true and false respectively.

Update The update function serves only to move the the camera rig to its starting position at (0, 510, 300), overlooking the plot. This needs to be done in the update, as the attributes determining position are not available at initialisation.

Tick The tick function triggers on every tick of the scene. It retrieves a reference to the camera rig and, if the trigger Boolean is true, moves the camera rig forward in the X axis at a set speed per tick, or if the grip pad Boolean is true, moves the camera rig backwards in the X axis at a set speed per tick.

3.3 Testing

3.3.1 Line Generation

Initially, data connection generation was performed manually at start-up. However, when each axis was creation, it triggered each sector to generate multiple times, as creation was treated the same as a positional change.

Regenerating the connection sectors multiple times on start-up resulted a much longer wait time for the visualisation to load. Therefore, the initial connection generation was removed, with zero repercussions.

Another problem encountered with the data connections was that initially each connection was its own 'a-datum' element, generated using a d3 select and given its individual coordinates as attributes. This resulted in major latency issues when moving the camera view, which were amplified tenfold under a VR headset, making it unusable. This was resolved by having one entity per connection sector, with that entity having a ’line__n’ component for each connection.

3.3.2 Benchmarking

Whilst the project does run on Firefox (version 59.0.3, current release as of writing), proven by its working behaviour on the non-VR in browser A-Frame viewer, moving the camera view introduces too much latency. As any latency is magnified tenfold when using a VR headset, due to its constant head-tracking changing the view, this is unacceptable. The scene loads correctly, but when the actual plot is looked at, the HMD either goes dark or suffers enough latency to cause motion sickness.

This resulted in a move to Google Chrome. Whilst Google Chrome does natively support WebVR, it needs to be enabled by setting several flags in the configuration. This is done by navigating to ’chrome://flags’, and setting the following flags to enabled: ’#enable-webvr’ and ’#enable-gamepad-extensions’. The following flags was also required on the machine used to test this project, however it may not prove necessary for all machines: ’#ignore-gpu-blacklist’. Initially only the the WebVR and A-Frame demos worked on Google Chrome, this was resolved by updated the A-Frame API to current version 0.8.0 from 0.6.1, the version at the start of development.

Moving to Google Chrome significantly improved the frame rate of the visualisation, allowing it to fully work within the HMD. However, having VR working resulted in different troubles. The mechanism for moving the camera at this point in time did not respond once VR mode was enabled, the camera was handled different and as such the mechanism needed to change. The camera itself was moved into a rig, linking together the camera, a placeholder for the user’s head position when in VR, and both controllers. Several different ideas for the mechanism itself were tested, including a teleportation system using the controllers to send forward an arch showing the destination, but it was decided that a more simpler solution would fit the project better. Since the camera only need to show the plot, and not the rest of the empty scene, the camera was to only scroll up and down the X axis using the trigger and grip-pad button to move forward and backwards, respectively. The view of the camera was locked facing -Z, positioned high up and looking down diagonally.

Several different browsers were tested at different versions to determine the optimum use case, resulting in this table:

Browser	Version	Result
Firefox Quantum	59.0.3	Too much latency to be viable
Google Chrome Stable	66.0.3359.139	Works as expected
Google Chrome Canary	68.0.3416.0	Works as expected

Chapter 4

Evaluation

There are numerous ways to evaluation the quality of a visualisation, including heuristic evaluations, the comparison against set principles; scenario evaluations, using different potential scenarios of usage to determine quality; or abstract task evaluations. As this project is intended to increase the immersion of a visualisation, this evaluation focuses more on the engagement aspect of the user experience.

To do so, I have implemented the VisEngage system, a self-assessment questionnaire centred around 11 different characteristics of user engagement; more detail is available in Chapter 2. The questionnaire features 22 questions, one per characteristic, using a 7-point Likert scale, from strongly agree (7) to strongly disagree (1). This gives a final engagement score out of 154 that provides a rough idea of how immersive the visualisation was.

To determine if these results were suitable for use in this evaluation, I measured the reliability, or internal consistency, of the questionnaire by calculating Cronbach's Alpha. The formula for Cronbach's Alpha is as follows:

$$\alpha = (\frac{k}{(k-1)}) * (1 - (\sum s_i^2)/s_t^2)$$

Where, k = number of items, or questions.
S_i = the standard deviation of i^{th} item.
S_t = the standard deviation of the sum score.
The resulting alpha value was 0.76 which, being between $0.6 \leq \alpha < 0.7$, is an acceptable internal consistency, determining that the questionnaire was reliable.

Alpha	Consistency
$0.9 \leq \alpha$	Excellent
$0.8 \leq \alpha < 0.9$	Good
$0.7 \leq \alpha < 0.8$	Acceptable
$0.6 \leq \alpha < 0.7$	Questionable
$0.5 \leq \alpha < 0.6$	Poor
$\alpha < 0.5$	Unacceptable

Due to the necessity of a HTC Vive HMD and set of controllers, the questionnaire could only be completed on site at the School of Computer Science's VisLab at Dean Street. This limited the availability of the questionnaire and as a result only five responses were obtained. While this low sample size does impact the reliability of the results, we are still able to gather a rough idea of the success of this project. The results of the questionnaire were as follows:

Question	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	SD
A	6	2	7	6	5	1.92
B	6	4	3	6	6	1.41
C	6	5	5	2	6	1.64
D	2	4	4	4	6	1.41
E	5	4	7	4	5	1.22
F	6	4	2	2	6	2
G	3	7	6	6	5	1.51
H	4	4	4	4	4	0
I	6	5	1	2	4	2.07
J	6	7	6	6	6	0.45
K	7	6	4	4	5	1.30
L	6	6	6	6	4	0.89
M	4	2	2	2	5	1.41
N	4	2	1	2	6	2
O	5	6	6	3	7	1.52
P	6	4	6	3	6	1.41
Q	7	6	2	4	7	2.17
R	5	5	5	3	7	1.41
S	7	6	7	3	5	1.67
T	6	6	6	4	5	0.89
U	2	6	5	4	4	1.48
V	7	1	2	6	7	2.88
Total	116	102	97	86	121	14.22

4.1 Results Breakdown

Participants were tasked to interact with the visualisation and use it to learn what they could about potential trends in the properties of different cars, the sample data provided. A blank copy of the questionnaire used is provided in Chapter A.

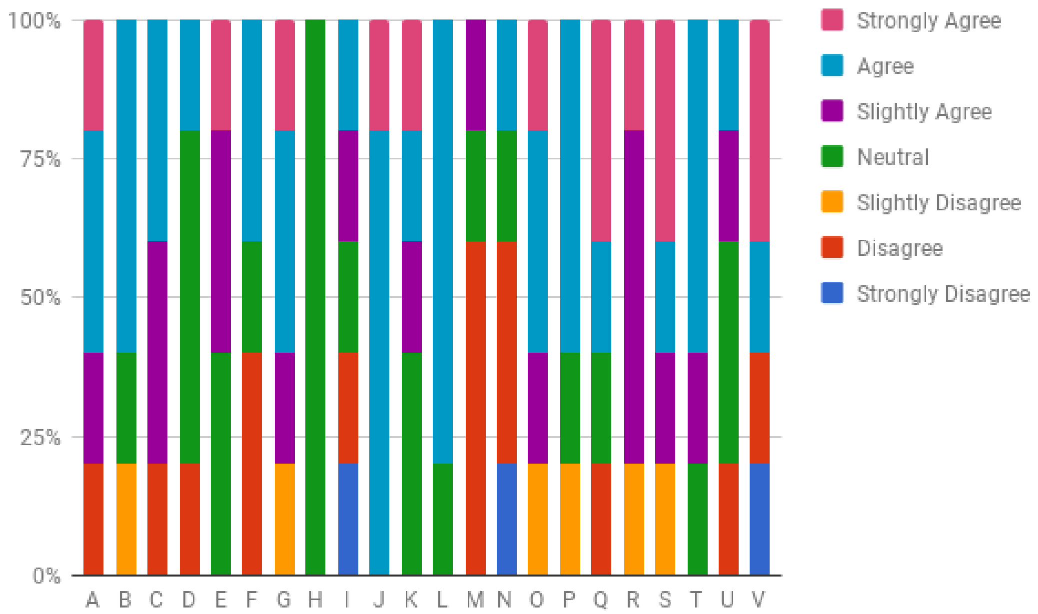


Figure 4.1: A stacked bar chart displaying the results of the questionnaire.

4.1.1 Aesthetics

The results showed that all but one of the participants found the plot to look pleasing, and most thought that the layout was clear and balanced.

4.1.2 Captivation

The results showed that all but on of the participants were absorbed in the visualisation, and felt unaware of their surroundings. Most were neutral on time passing quickly, with one agreeing and one disagreeing.

4.1.3 Challenge

The average response to enjoying and accepting the challenge presented by the plot was slightly above neutral, with one response strongly agreeing. Participants differed on whether the plot caused them to think carefully, deeply, or reflectively, with two for both disagreeing and agreeing, and one neutral.

4.1.4 Control

Most participants felt that the functions and features worked as expected, with only one use slightly disagreeing.

4.1.5 Discovery

Participants differed on whether they had learned something new from the visualisation, as all responses were different, with the average shifted towards disagreeing. All participants agreed that they figured out how to use the plot along the way, with four agreeing responses and one strongly agreeing.

4.1.6 Exploration

Most participants felt that they were moving in or through the interactive plot to learn about its content, with two feeling neutral. All agreed that they were exploring its features and content in a gradual fashion, with one participant neutral.

4.1.7 Creativity

Most disagreed that they found themselves imagining thing not directly related to the chart, with one person slightly agreeing and one neutral. Most disagreed that they found themselves generating new ideas, though one agreed and one was neutral.

4.1.8 Attention

Most participants agreed that they found themselves concentrating on specific aspects or features of the plot, though one slightly disagreed. Most were also in agreement that they had to pay attention to multiple things at the same time.

4.1.9 Interest

Most felt that the content of the plot was interesting, with two strongly agreeing, and only one disagreeing. Most also felt that the features and interaction of the plot were interesting, with one slightly disagreeing.

4.1.10 Novelty

Most participants felt that the look and feel visualisation was novel and fresh, with two strongly agreeing, and only one slightly disagreeing. Most also agreed that the features and interactions provided were novel and fresh, with one feeling neutral.

4.1.11 Autotelism

Results were varied on whether participants experienced enjoyment from the visualisation itself, more feeling neutral with the average response shifted towards agreement. Responses were further varied on whether the participant would want to use the visualisation of their own accord, rather than being required or encouraged, with two strongly agreeing, one agreeing, one disagreeing, and one strongly disagreeing.

4.2 Discussion

Overall the results of the questionnaire were positive, and the engagement scores were all above 77, showing that the visualisation was engaging, if not particularly strong.

Immersive Analytics is a relatively new and burgeoning subject area, and the standards and resources surrounding it are open and incomplete, especially when applying its ideas towards the principles of web development. Beta versions of browsers were necessary, and under-the-hood flags had to be set, showing that while web VR development is rapidly expanding, it is still a new concept. This presented various challenges to the progress of this project, as existing work was sparse and necessary APIs still in pre-release development. A hosting solution was needed, as the use of VR was restricted to HTTPS connections to prevent vulnerabilities as its access is expanded and developed. Thankfully, GitHub provided both the repository to store the work, and the HTTPS access through the GitHub Pages system. The result was a challenging, if rewarding, dissertation project.

Chapter 5

Profession, Social, Technological, Legal and Ethical (PeSTLE) Implications

There are minimal implications for a visualisation project of this sort. Professional considerations are merely that the licensing conditions of the various APIs were acknowledged and met. A-Frame is released under the MIT license, requiring only that the copyright and permission notice be included in the project. D3 states similar conditions under the BSD 3-Clause license, with the added clause that the name of its author and contributors must not be used to endorse derived products. This extra clause has been met as well.

Copyright protection may result in potential problems, were restricted data sets to be used in forming the parallel coordinate plot. While the project is dynamic in that any comma separated value file of any proportion and dimensions can be used to generate the plot, only a public sample data set of car properties is included.

Chapter 6

Conclusions and Future Work

Time permitting, there are various changes and improvements that would be beneficial to this project, some mere ideas, others more substantial.

6.1 Customisation

The plot itself could be improved by allowing more properties of it to be customisable by the user. Axes should be able to be coloured individually, with data connections also having different colours. This would allow the plot to be more easily readable, as different sections would stand out better and possible be easier on the eyes.

The axes should allow rotation to allow the plot to be laid flat, resulting in a top-down view of the scene. Axis rotation would also necessitate more complex camera movement, using more than just the X axis.

A reference line could be added that shows the original position of the axes on generation. This would add more perspective to the scene, allowing the user to more easily get their bearings against a completely white floor plane.

6.2 In Depth Analysis

Due to limitations in the plot, individual datum are difficult to access. This is because each line is stored as a component in its sector's element to prevent latency problems. The 'n' section of the 'line__n' component should act as a unique identifier for each record or datum in the data set. Accessing one full polyline in the plot would require referencing each 'line__n' component with the correct 'n' identifier in every sector element of the plot. This may end up being more resource intensive then restructuring the plot. Another issue that would arise would be selecting individual polylines with the raycaster, as the target would potentially be too small. The 'line__n' component could be replaced with a custom solution that makes use of meshes with box geometry, though this may add more strain on the latency and increased line thickness would result in more of the plot blurring together. Selecting polylines with a click and drag technique would solve this issue, but prevent the selection of individual polylines, instead only allowing a grouping.

Accessing the polyline of a datum would allow better analysis of the plot as a whole. Colours could be set to make and individual polyline, or grouping of polylines, stand out against the rest. Readings could be given and displayed above the controller of the that section's data from the source set.

References

[1] A-Frame. (). A-frame - make webvr, [Online]. Available: https://aframe.io (visited on 3rd May 2018) (p. 15).

[2] A. Artaud, *The Theater and Its Double*, ser. Evergreen Original E-127. Grove Press, 1958, ISBN: 9780802150301. [Online]. Available: https://books.google.co.uk/books?id=bmf8CMzu3kIC (p. 6).

[3] N. G. Belmonte. (). Engineering intelligence through data visualization at uber, [Online]. Available: https://eng.uber.com/data-viz-intel/ (visited on 3rd May 2018) (p. 12).

[4] C-SPAN. (). Vpl research | c-span.org, [Online]. Available: https://www.c-span.org/organization/?112710/VPL-Research (visited on 26th Apr. 2018) (p. 6).

[5] X. Chen. (). Engineering uber's self-driving car visualization platform for the web, [Online]. Available: https://eng.uber.com/atg-dataviz/ (visited on 3rd May 2018) (p. 12).

[6] D. Community. (). D3.js - data-driven documents, [Online]. Available: d3js.org (visited on 3rd May 2018) (p. 15).

[7] B. Delaney, *Sex, Drugs and Tessellation: The Truth About Virtual Reality, as Revealed in the Pages of CyberEdge Journal*. CyberEdge Information Services, 2014. [Online]. Available: https://books.google.co.uk/books/about/Sex_Drugs_and_Tessellation.html?id=4aelBQAAQBAJ&redir_esc=y (p. 7).

[8] R. K. Elliott, 'Socrates and plato's cave', vol. Vol 58, Issue 1-4, pp. 137–157, Sep. 2009 (p. 10).

[9] A. S. .-.-. Engadget. (). How htc and valve built the vive, [Online]. Available: https://www.engadget.com/2016/03/18/htc-vive-an-oral-history/ (visited on 2nd May 2018) (p. 9).

[10] S. S. Fisher, M. McGreevy, J. Humphries and W. Robinett, ‘Virtual environment display system’, in *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, ser. I3D ’86, Chapel Hill, North Carolina, USA: ACM, 1987, pp. 77–87, ISBN: 0-89791-228-4. DOI: 10.1145/319120.319127. [Online]. Available: http://doi.acm.org/10.1145/319120.319127 (p. 8).

[11] V. Focus. (). Palmer luckey explains oculus rift’s constellation tracking and fabric, [Online]. Available: https://www.vrfocus.com/2015/06/palmer-luckey-explains-oculus-rifts-constellation-tracking-and-fabric/ (visited on 2nd May 2018) (p. 9).

[12] D. Goldman. (). Google unveils ’project glass’ virtual-reality glasses, [Online]. Available: http://money.cnn.com/2012/04/04/technology/google-project-glass/ (visited on 3rd May 2018) (p. 11).

[13] E. Government. (). Welcome | european union open data portal, [Online]. Available: https://www.data.europa.eu/euodp/data (visited on 18th Apr. 2018) (p. 2).

[14] U. Government. (). Find open data - data.gov.uk, [Online]. Available: https://www.data.gov.uk (visited on 18th Apr. 2018) (p. 2).

[15] U. Government. (2004). Data.gov, [Online]. Available: https://www.data.gov (visited on 18th Apr. 2018) (p. 2).

[16] K. Guttag. (). Proof that google glass uses a himax lcos microdisplay, [Online]. Available: https://seekingalpha.com/article/1504292-proof-that-google-glass-uses-a-himax-lcos-microdisplay (visited on 3rd May 2018) (p. 11).

[17] J. Hempel. (). Project hololens: Our exclusive hands-on with microsoft’s holographic goggles, [Online]. Available: https://www.wired.com/2015/01/microsoft-hands-on/ (visited on 3rd May 2018) (p. 11).

[18] K. Horowitz. (). Sega vr: Great idea or wishful thinking?, [Online]. Available: https://web.archive.org/web/20100114191355/http://sega-16.com/feature_page.php?id=5&title=Sega%5C%20VR:%5C%20Great%5C%20Idea%5C%20or%5C%20Wishful%5C%20Thinking? (visited on 2nd May 2018) (p. 8).

[19] Y.-H. Hung and P. Parsons, ‘Assessing user engagement in information visualization’, in *Proceedings of the 2017 CHI Conference Extended*

Abstracts on Human Factors in Computing Systems, ser. CHI EA '17, Denver, Colorado, USA: ACM, 2017, pp. 1708–1717, ISBN: 978-1-4503-4656-6. DOI: 10.1145/3027063.3053113. [Online]. Available: http://doi.acm.org/10.1145/3027063.3053113 (p. 13).

[20] R. Kalawsky, *The Science of Virtual Reality and Virtual Environments: A Technical, Scientific and Engineering Reference on Virtual Environments*. Addison-Wesley, 1993, ISBN: 9780201631715. [Online]. Available: https://books.google.co.uk/books?id=n-BQAAAAMAAJ (pp. 6, 10).

[21] T. H. B. Line. (). Controversy grows over google's glass project, [Online]. Available: https://www.thehindubusinessline.com/info-tech/controversy-grows-over-googles-glass-project/article20595712.ece (visited on 3rd May 2018) (p. 11).

[22] Microsoft. (). Introducing the microsoft hololens development edition, [Online]. Available: http://www.microsoft.com/microsoft-hololens/en-us/development-edition (visited on 3rd May 2018) (p. 11).

[23] P. Milgram and F. Kishino, 'A taxonomy of mixed reality visual displays', vol. vol. E77-D, no. 12, pp. 1321–1329, Dec. 1994 (p. 4).

[24] D. J. Sturman and D. Zeltzer, 'A survey of glove-based input', *IEEE Computer Graphics and Applications*, vol. 14, no. 1, pp. 30–39, Jan. 1994, ISSN: 0272-1716. DOI: 10.1109/38.250916 (p. 7).

[25] I. E. Sutherland, 'The ultimate display', in *Information processing 1965: proceedings of IFIP Congress 65. Vol. 2*, ser. IFIP '65, New York City, 1965, pp. 506–508. [Online]. Available: http://www8.informatik.umu.se/~jwworth/The%5C%20Ultimate%5C%20Display.pdf (p. 5).

[26] ——, 'A head-mounted three dimensional display', in *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, ser. AFIPS '68 (Fall, part I), San Francisco, California: ACM, 1968, pp. 757–764. DOI: 10.1145/1476589.1476686. [Online]. Available: http://doi.acm.org/10.1145/1476589.1476686 (p. 5).

[27] G. Tavares. (). Webgl fundamentals, [Online]. Available: http://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals/ (visited on 3rd May 2018) (p. 14).

[28] S. M. .-.-. A. Technica. (). Htc vive pre impressions: A great vr system has only gotten better, [Online]. Available: https://arstechnica.com/gaming/2016/01/htc-vive-pre-impressions-a-great-vr-system-has-only-gotten-better/ (visited on 2nd May 2018) (p. 10).

[29] C. W. .-.-. T. Verge. (). New oculus rift dev kit uses the front of a galaxy note 3 as its screen, [Online]. Available: https://www.theverge.com/2014/7/31/5956589/new-oculus-dev-kit-uses-front-of-galaxy-note-3-for-display (visited on 2nd May 2018) (p. 9).

[30] O. VR. (). First look at the rift, shipping q1 2016, [Online]. Available: https://www.oculus.com/blog/first-look-at-the-rift-shipping-q1-2016/ (visited on 2nd May 2018) (p. 9).

[31] ——, (). Update on developer kit technology, shipping details, [Online]. Available: https://www.oculus.com/blog/update-on-developer-kit-technology-shipping-details/ (visited on 2nd May 2018) (p. 8).

[32] I. Web. (). Webxr device api explained, [Online]. Available: https://github.com/immersive-web/webxr/blob/master/explainer.md (visited on 3rd May 2018) (p. 14).

[33] T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson and Y. Harvill, 'A hand gesture interface device', *SIGCHI Bull.*, vol. 17, no. SI, pp. 189–192, May 1986, ISSN: 0736-6906. DOI: 10.1145/30851.275628. [Online]. Available: http://doi.acm.org/10.1145/30851.275628 (p. 7).

Appendix A

Code and Other Resources

A.1 vrplot.html

```
<!DOCTYPE html>
<html>
<meta charset="utf-8">
  <head>
    <script src="https://d3js.org/d3.v3.min.js"></script>
    <script src="https://aframe.io/releases/0.8.0/aframe.min.js
        "></script>
    <script src="generatescene.js"></script>
  </head>

  <body style='margin : 0px; overflow: hidden;'>
    <a-scene>
      <a-entity id="cameraRig" rotation="0␣180␣0">
        <!-- camera -->
        <a-entity id="head" camera wasd-controls look-controls>
            </a-entity>
        <!-- hand controls -->
        <a-entity id="left-hand" movement-listener="cameraRig:␣
            #cameraRig" hand-controls="left">
          <a-text value="Squeeze␣hand␣to␣move␣backward␣Pull␣
              trigger␣to␣move␣forward" position="0.14␣0.2␣0"
```

```
            rotation="0␣0␣-90" wrap-count="27" scale="0.1␣0.1␣
            0.1"></a-text>
        </a-entity>
        <a-entity id="right-hand" controller-listener laser-
            controls="hand:␣right" line="color:␣red">
          <a-text value="Push␣the␣middle␣pad␣to␣select␣axis␣
            Push␣again␣to␣choose␣new␣position" position="-0.2␣
            0.1␣0" rotation="0␣0␣0" wrap-count="30" scale="0.1
            ␣0.1␣0.1"></a-text>
        </a-entity>
      </a-entity>
      <a-sky color='#4CC3D9' fuse timeout='100'></a-sky>
      <a-plane color='#CCC' height="2000" width="2000" rotation
        ="-90␣0␣0"></a-plane>
    </a-scene>
  </body>
</html>
```

A.2 generatescene.js

```
var width = 960,
  height = 500;

var x = d3.scale.ordinal()
  .rangePoints([0, width], 1),
  y = {},
  dragging = {};

var line = d3.svg.line();
var background, foreground;

var csvData;
var names;
```

```
var offsetx = 500;
var zOffset = -100;

var selectedElement;
var ghostElement;
var holding = false;
var previousColour;

AFRAME.registerComponent('controller-listener', {
  init: function() {

    // Happens when both trigger is pressed and laser is
       intersecting
    this.el.addEventListener('click', function(evt) {
      // If we are moving an element
      if (holding) {
        // Replace old element with new element
        var newCoords = evt.detail.intersection.point;

        // Move the select element to the new position
        selectedElement.setAttribute('positionX', newCoords.x);
        selectedElement.setAttribute('positionZ', newCoords.z);
        selectedElement.setAttribute('colour', previousColour);

        holding = false;
        selectedElement = undefined;
      } else {
        // Store this element
        selectedElement = evt.detail.intersectedEl;
        var axis = selectedElement.getAttribute('axis');

        if (!(typeof axis === "undefined")) {
```

```
          // Axis

          previousColour = axis.colour;
          holding = true;
          // Give visual feedback
          selectedElement.setAttribute('colour', 'orange');
        }
      }
    });
  }
});

AFRAME.registerComponent('movement-listener', {
  schema: {
    cameraRig: {
      type: 'selector'
    },
    speed: {
      type: 'int',
      default: 2
    }
  },

  init: function() {
    // Record if buttons are active
    this.trigger = false;
    this.grippad = false;
    var el = this.el;

    this.onTriggerUp = this.onTriggerUp.bind(this);
    this.onGripUp = this.onGripUp.bind(this);
    this.onGripDown = this.onGripDown.bind(this);
    this.onTriggerDown = this.onTriggerDown.bind(this);
```

```
    // Set the event listeners for each button event
    el.addEventListener('triggerup', this.onTriggerUp);
    el.addEventListener('triggerdown', this.onTriggerDown);
    el.addEventListener('gripup', this.onGripUp);
    el.addEventListener('gripdown', this.onGripDown);
  },

  update: function() {
    const rig = this.data.cameraRig || this.el.sceneEl.camera.
        el;
    var currentPosition = rig.getAttribute('position');
    rig.setAttribute('position', {x: currentPosition.x + this.
        data.speed, y: 510, z: 300});
  },

  tick: function(time, delta) {
    var currentPosition;
    var data = this.data;
    var el = this.el;
    var speed = this.speed;

    // Retrieve the camera rig, or camera element
    const rig = data.cameraRig || el.sceneEl.camera.el;
    // If the trigger is active
    if (this.trigger) {
      // Move position forward
      currentPosition = rig.getAttribute('position');
      rig.setAttribute('position', {x: currentPosition.x + data
          .speed, y: 510, z: 300});
    } else if (this.grippad) {
      // If grippad is active, move backwards
      currentPosition = rig.getAttribute('position');
```

```
      rig.setAttribute('position', {x: currentPosition.x - data
          .speed, y: 510, z: 300});
    }
  },

  onTriggerDown: function(event) {
    this.trigger = true;
  },

  onTriggerUp: function(event) {
    this.trigger = false;
  },

  onGripDown: function(event) {
    this.grippad = true;
  },

  onGripUp: function(event) {
    this.grippad = false;
  }
});

d3.csv("cars.csv", function(cars) {
  // Store data
  csvData = cars;
  csvData.map(function(d) {
    return +d;
  });
  // Generate axes
  // Delete old axis (if they exist)
  d3.select("a-scene").selectAll("a-datum").remove();
  // Generate column position scale
  names = d3.keys(cars[0]).filter(function(column_name) {
```

```
  return column_name != "name";
})
var xScale = d3.scale.linear();
xScale.range([0, width]);
xScale.domain([0, (names.length - 1)]);

d3.select("a-scene").selectAll("a-axis")
  .data(names)
  .enter()
  .append("a-axis")
  .each(function(datum, index) {
    // Select itself
    var entity = d3.select(this);
    var boundaries = d3.extent(cars, function(max_values) {
      return +max_values[datum];
    });

    console.log("Creating axis at x=" + xScale(index) + "
       using " + index + " for " + datum);
    console.log("Boundardies: " + boundaries);

    // Set attributes
    entity.attr("name", datum);
    entity.attr("positionorder", index);
    entity.attr("positionx", xScale(index));
    entity.attr("positionz", zOffset);
    entity.attr("height", height);
    entity.attr("minvalue", boundaries[0]);
    entity.attr("maxvalue", boundaries[1]);

    var text = entity.append("a-text");
    text.attr("value", datum);
    text.attr("align", "center");
```

```
    });
});

function generateLines(sector, lineColour) {
  if (sector < 0) {
    // No section here
    return 0;
  }

  // eg section 1: axes 1 to 2
  // Retrieve axes of section
  var leftAxis = d3.select("a-scene").select("a-axis[
      positionorder='" + sector + "']");
  var rightAxis = d3.select("a-scene").select("a-axis[
      positionorder='" + (sector + 1) + "']");

  // Check if at right edge
  if (rightAxis.empty()) {
    // No section here
    return 0;
  }

  // Generate y-scale for left axis
  var leftYScale = d3.scale.linear();
  leftYScale.domain([leftAxis.attr("minvalue"), leftAxis.attr("
      maxvalue")]);
  leftYScale.range([leftAxis.attr("height"), 0]);

  // Generate y-scale for right axis
  var rightYScale = d3.scale.linear();
  rightYScale.domain([rightAxis.attr("minvalue"), rightAxis.
      attr("maxvalue")]);
  rightYScale.range([rightAxis.attr("height"), 0]);
```

```
  // Remove old lines
  d3.select("a-scene").selectAll("a-datum[sector='" + sector +
      "']").remove();

  console.log("Generating lines for sector: " + sector);

  // Create a container element for these connections
  var entity = d3.select("a-scene").append("a-datum");
  entity.attr("sector", sector);
  var index = 1;
  var value;

  // For each connection
  for (let datum of csvData) {
    // Calculate and set its coordinates from the scales
    value = "start: " + leftAxis.attr("positionx") + ", " +
        leftYScale(datum[leftAxis.attr("name")]) + ", " +
        leftAxis.attr("positionz");
    value += "; end: " + rightAxis.attr("positionx") + ", " +
        rightYScale(datum[rightAxis.attr("name")]) + ", " +
        rightAxis.attr("positionz");
    // Set the line colour
    value += "; color: " + lineColour;
    // Create the new line
    entity.attr("line__" + index, value);
    index++;
  }
}

/**
Represents one axis
*/
```

```
AFRAME.registerComponent('axis', {
  schema: {
    colour: {
      default: '#FFF'
    },
    linecolour: {
      default: 'black'
    },
    name: {
      type: 'string',
      default: ''
    },
    positionOrder: {
      type: 'int',
      default: 1
    },
    positionX: {
      type: 'number',
      default: 1
    },
    positionZ: {
      type: 'number',
      default: 1
    },
    height: {
      type: 'number',
      default: height
    },
    maxValue: {
      type: 'number',
      default: height
    },
    minValue: {
```

```
    type: 'number',
    default: 0
  }
},

init: function() {
  var data = this.data;
  var geometry;
  var material;

  // Generate a new mesh material
  material = this.material = new THREE.MeshBasicMaterial({
    color: data.colour,
    opacity: 1,
    transparent: 0,
    visible: true
  });

  // Buffer new geometry for the mesh
  geometry = this.geometry = new THREE.BoxBufferGeometry(4,
     data.height, 4, 1, 1, 1);

  // Create the mesh and set its position
  this.mesh = new THREE.Mesh(geometry, material);
  this.mesh.position.set(data.positionX, data.height / 2,
     data.positionZ);

  // Apply the mesh to the element
  this.el.setObject3D('mesh', this.mesh);
},

update: function(oldData) {
  // On property update (and after init)
```

```
var data = this.data; // Property values
var geometry = this.geometry;
var geoNeedsUpdate = false;
var material = this.material;
var mesh = this.mesh;
var el = this.el;
var text = el.children[0];

// Geometry change
if (data.positionX !== oldData.positionX ||
  data.positionZ !== oldData.positionZ) {
  // Check for overlap
  var prevAxis = d3.select("a-scene").select("a-axis[
     positionorder='" + (data.positionOrder - 1) + "']");
  var nextAxis = d3.select("a-scene").select("a-axis[
     positionorder='" + (data.positionOrder + 1) + "']");

  // Check for order changes
  if (!prevAxis.empty() && (data.positionX < prevAxis.attr
     ("positionx"))) {
    // Axis has been moved down the list
    // Swap prevAxis' positionOrder
    prevAxis.attr("positionorder", data.positionOrder);
    // Decrement positionOrder
    var thisAxis = d3.select("a-scene").select("a-axis[name
       ='" + data.name + "']");
    thisAxis.attr("positionorder", data.positionOrder - 1);
    data.positionOrder--;
  } else if (!nextAxis.empty() && (data.positionX >
     nextAxis.attr("positionx"))) {
    // Axis has been moved up the list
    // Swap nextAxis' positionOrder
    nextAxis.attr("positionorder", data.positionOrder);
```

```
    // Increment positionOrder
    var thisAxis = d3.select("a-scene").select("a-axis[name
        ='" + data.name + "']");
    thisAxis.attr("positionorder", data.positionOrder + 1);
    data.positionOrder++;
  }

  // Update lines
  if (data.positionOrder == 0) {
    // Start
    generateLines(0, data.linecolour);
    generateLines(1, data.linecolour);
  } else if (data.positionOrder == names.length - 1) {
    // End
    generateLines(data.positionOrder - 1, data.linecolour);
    generateLines(data.positionOrder - 2, data.linecolour);
  } else {
    // Middle
    for (i = (data.positionOrder - 2); i < data.
        positionOrder + 3; i++) {
      generateLines(i, data.linecolour);
    }
  }
}

// Update text
//text.object3D.position.set(data.positionX, data.height +
    10, data.positionZ);
//text.object3D.scale.set(40, 40, 40);
text.setAttribute("name", data.name);
//text.object3D.updateMatrixWorld();
text.setAttribute('position', {x: data.positionX, y: data.
    height + 10, z: data.positionZ});
```

```
    text.setAttribute('scale', {x: 40, y: 40, z: 40});

    // Update mesh position and colour
    mesh.position.set(data.positionX, data.height / 2, data.
        positionZ);
    material.color.set(data.colour);
  },

  remove: function() {
    this.el.removeObject3D('mesh');
  }
})

/**
Represents one axis
*/
AFRAME.registerComponent('datum', {
  schema: {
    colour: {
      default: 'red'
    },
    sector: {
      type: 'int',
      default: 0
    }
  }
})

AFRAME.registerPrimitive('a-axis', {
  // Attaches 'axis' to 'a-axis'
  defaultComponents: {
    axis: {}
  },
```

```
  mappings: {
    colour: 'axis.colour',
    name: 'axis.name',
    positionorder: 'axis.positionOrder',
    positionx: 'axis.positionX',
    positionz: 'axis.positionZ',
    height: 'axis.height',
    maxvalue: 'axis.maxValue',
    minvalue: 'axis.minValue',
    linecolour: 'axis.linecolour'
  }
})

AFRAME.registerPrimitive('a-datum', {
  // Attaches 'axis' to 'a-axis'
  defaultComponents: {
    datum: {}
  },

  mappings: {
    colour: 'datum.colour',
    sector: 'datum.sector'
  }
})
```

VisEngage Project Questionnaire

Questionnaire to be completed after testing the A-Frame Parallel Coordinate Plot visualisation, using the VisEngage user engagement assessment system. Thank you for your participation.

*Required

Please enter your name. *

Your answer

A. While using this interactive plot, I found its look and feel to be pleasing. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

B. The layout of this interactive plot is clear and balanced. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

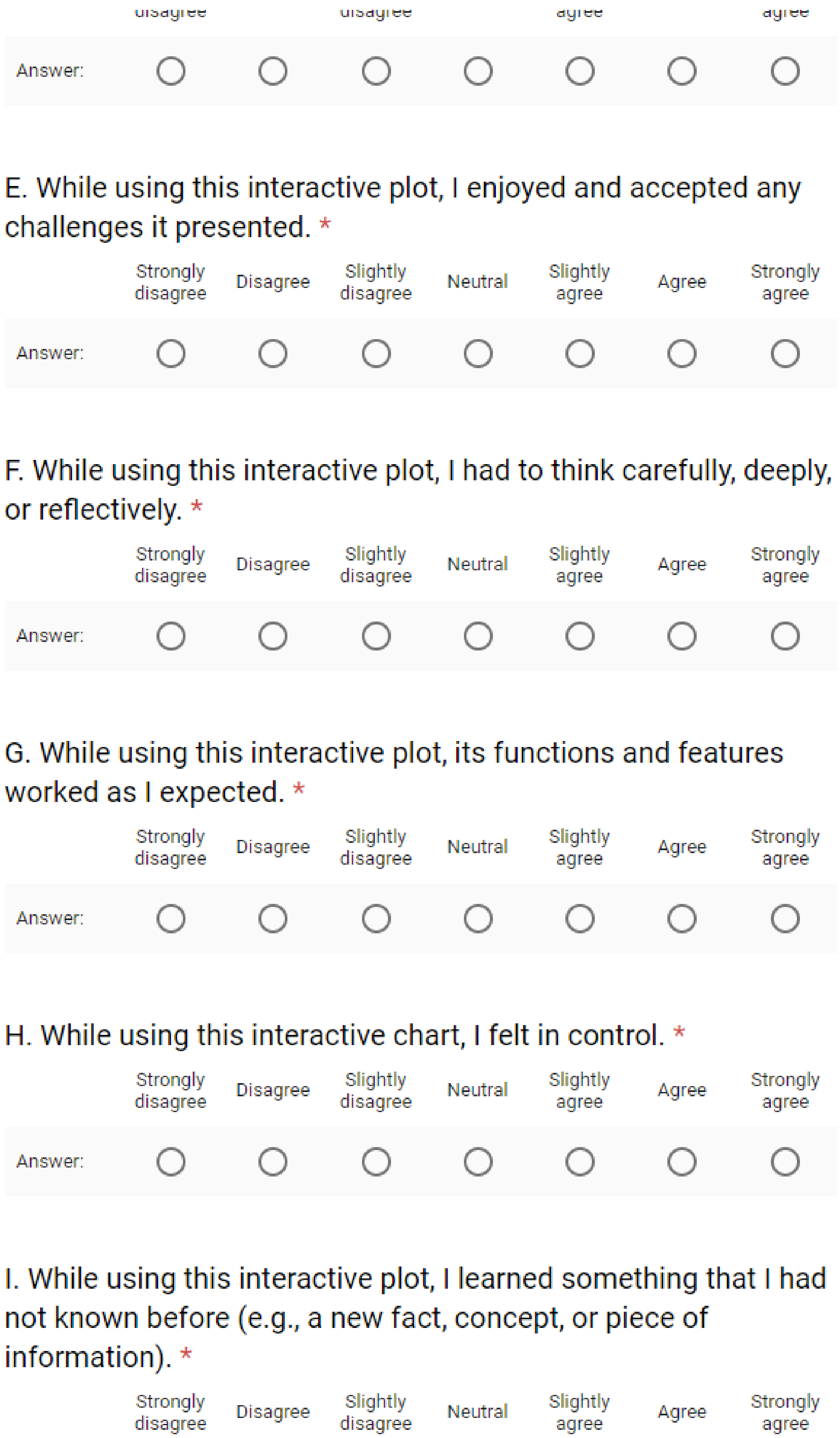
C. While using this interactive plot, I felt absorbed to the extent that I was not aware of my surroundings. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

D. While using this interactive plot, time seemed to pass quickly. *

	Strongly	Disagree	Slightly	Neutral	Slightly	Agree	Strongly

Figure A.1: Partial screenshot of the questionnaire, found at https://goo.gl/forms/0GjkHZBhfn5YCAXC2



	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

E. While using this interactive plot, I enjoyed and accepted any challenges it presented. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

F. While using this interactive plot, I had to think carefully, deeply, or reflectively. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

G. While using this interactive plot, its functions and features worked as I expected. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

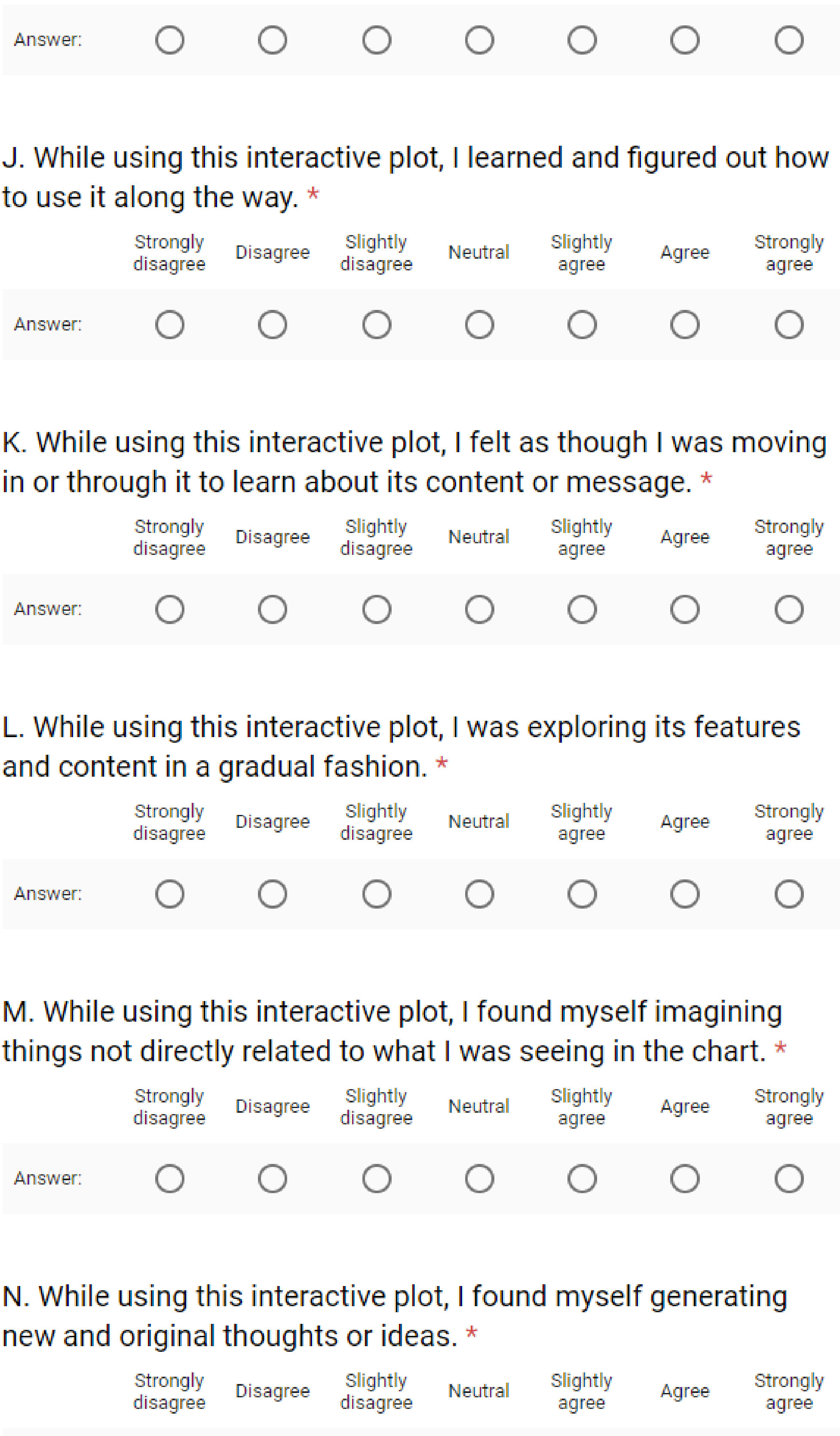
H. While using this interactive chart, I felt in control. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

I. While using this interactive plot, I learned something that I had not known before (e.g., a new fact, concept, or piece of information). *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree

Figure A.2: Partial screenshot of the questionnaire, found at https://goo.gl/forms/0GjkHZBhfn5YCAXC2



Answer:	○	○	○	○	○	○	○

J. While using this interactive plot, I learned and figured out how to use it along the way. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

K. While using this interactive plot, I felt as though I was moving in or through it to learn about its content or message. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

L. While using this interactive plot, I was exploring its features and content in a gradual fashion. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

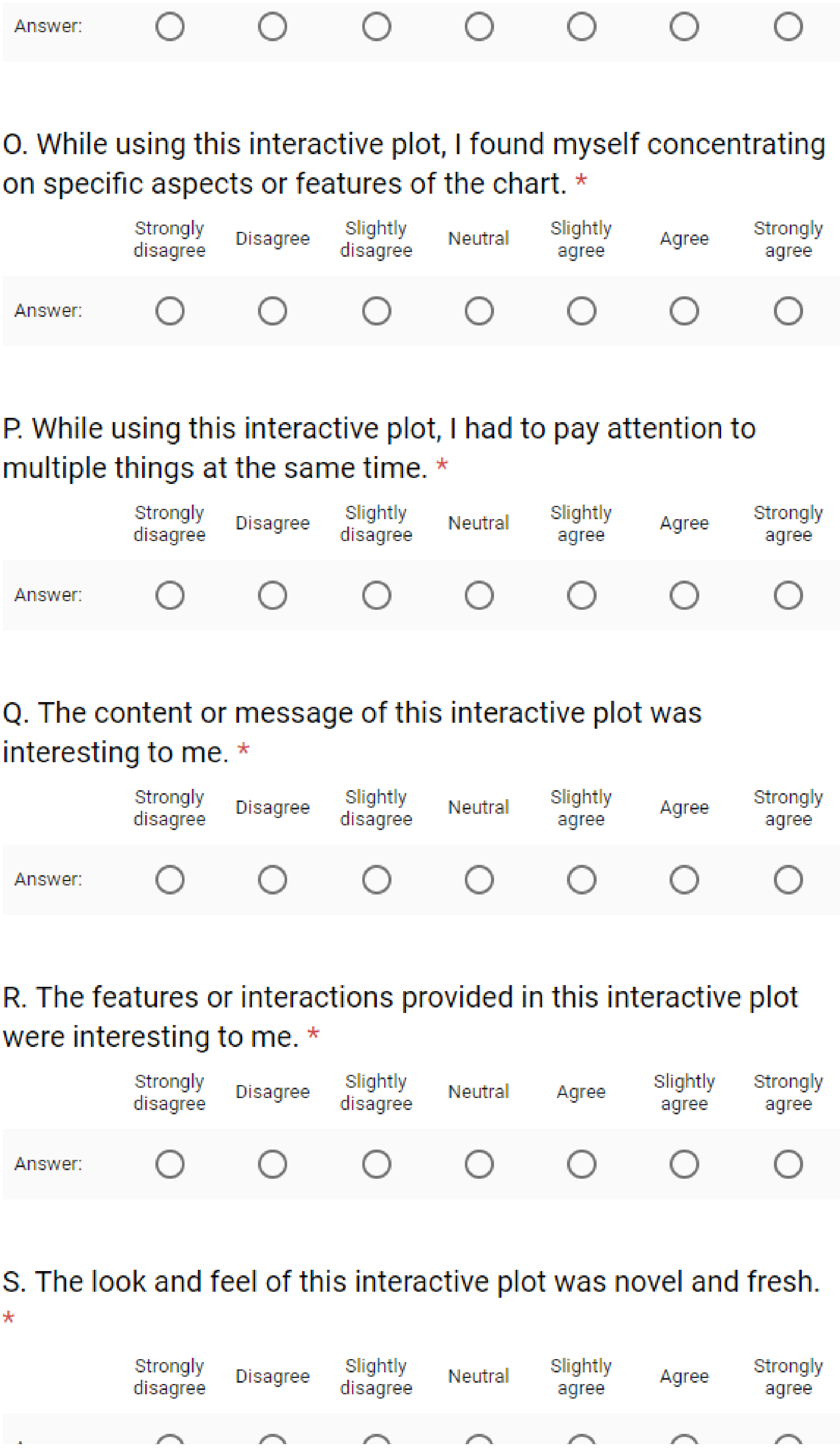
M. While using this interactive plot, I found myself imagining things not directly related to what I was seeing in the chart. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

N. While using this interactive plot, I found myself generating new and original thoughts or ideas. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree

Figure A.3: Partial screenshot of the questionnaire, found at https://goo.gl/forms/0GjkHZBhfn5YCAXC2



Answer:							

O. While using this interactive plot, I found myself concentrating on specific aspects or features of the chart. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:							

P. While using this interactive plot, I had to pay attention to multiple things at the same time. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:							

Q. The content or message of this interactive plot was interesting to me. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:							

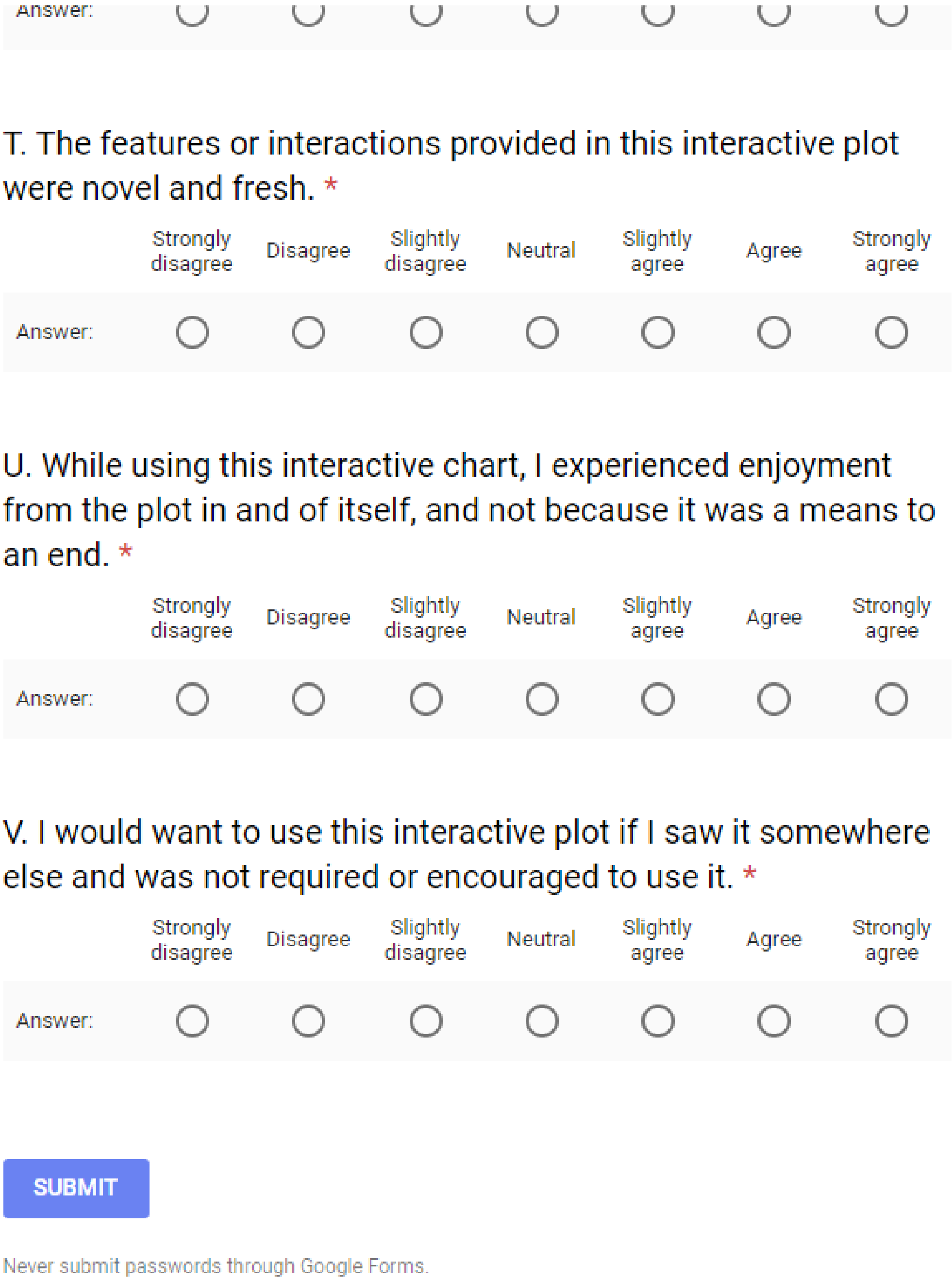
R. The features or interactions provided in this interactive plot were interesting to me. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Agree	Slightly agree	Strongly agree
Answer:							

S. The look and feel of this interactive plot was novel and fresh. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree

Figure A.4: Partial screenshot of the questionnaire, found at https://goo.gl/forms/0GjkHZBhfn5YCAXC2



Answer:

T. The features or interactions provided in this interactive plot were novel and fresh. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

U. While using this interactive chart, I experienced enjoyment from the plot in and of itself, and not because it was a means to an end. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

V. I would want to use this interactive plot if I saw it somewhere else and was not required or encouraged to use it. *

	Strongly disagree	Disagree	Slightly disagree	Neutral	Slightly agree	Agree	Strongly agree
Answer:	○	○	○	○	○	○	○

SUBMIT

Never submit passwords through Google Forms.

Figure A.5: Partial screenshot of the questionnaire, found at https://goo.gl/forms/0GjkHZBhfn5YCAXC2

Appendix B

Project Poster

Author:
Joshua Henderson
eeu67d@bangor.ac.uk

Immersive Analytics: Parallel Coordinate Plots in Virtual Reality

Supervisor:
Dr. Panagiotis D. Ritsos
p.ritsos@bangor.ac.uk

PRIFYSGOL BANGOR UNIVERSITY

Introduction

This project aims to create an immersive, virtual environment for the dynamic visualisation of high-dimensional multivariate data.

It does this by generating an interactive parallel coordinate plot using the A-Frame virtual reality web framework. This then allows it to be displayed platform independent through a current-generation web browser with WebVR capability, on an Oculus Rift, HTC Vive, or other compatible head-mounted display.

Interactivity will be achieved either through the Oculus Touch platform, or the Vive Controllers, dependent on what device is connected.

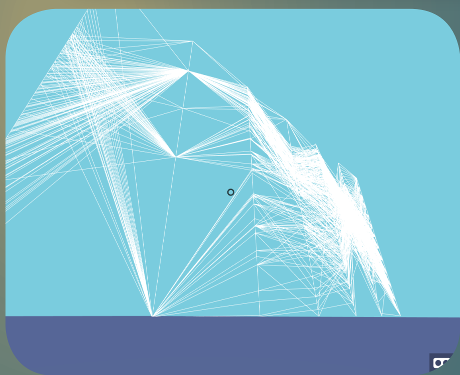
Frameworks

A-Frame describes itself as an 'open-source web framework for building virtual reality experiences'. It builds upon Three.js and WebVR to handle control of the head-mounted display, allowing the developer to focus on creating their scenes using HTML tags in the Document Object Model.

Three.js is a cross-platform JavaScript library that uses WebGL to allow the creation of GPU-accelerated 3D animations that are not dependent on browser plugins.

WebGL is cross-platform JavaScript API that allows the rendering of interactive 2D and 3D graphics within compatible web browsers. using OpenGL and the OpenGL Shading Language (GLSL).

D3.js (Data-Driven Documents) is a cross-platform JavaScript library that allows the creation of dynamic, interactive data visualisations in the browser, with programmatic control of the HTML DOM.



Immersive Analytics

Immersive Analytics is a relatively new initiative that combines the disciplines of data science, mixed reality, and 3D interfaces to explore and develop new interaction and visualisation technologies to expand the realm of data analytics and support further analytical reasoning and decision techniques.

Current technologies within this initiative include:

- The CAVE2 virtual reality collaboration system, a room-scale environment that combines scalable-resolution display walls and virtual reality systems, creating a seamless 2D/3D, information-rich simulation.
- The AlloSphere, a three-story sphere that combines twenty-six projectors and fifty-five speakers to fully immerse those on it's bridge in scientific simulations, data visualisations, and artistic content.
- The Uber Self-Driving Car Visualisation Platform, a web-based project that unites data from sources such as preprocessed maps, and runtime-generated vehicle logs to recreate the context around a trip and understand and develop the decisions an autonomous vehicle must make.

Development

Development of the main stage of the project is nearing completion, with working JavaScript code to generate the lines for the axes and plot points. The next objective to implement the design for visualisation interactivity, including axes positioning to take advantage of the current work that actively determines the order for each dimension and regenerates data connections accordingly.

Further interactivity will include customisation, such as individual axis size colour, and system to give focus and indication to selected data points, with a tool attached to the virtual representation of the controller that can display the measurement information the desired data point.

A final stage of development will then commence to tidy up the existing work, including a visual redesign, code refactoring, and completion of any potential missed documentation.