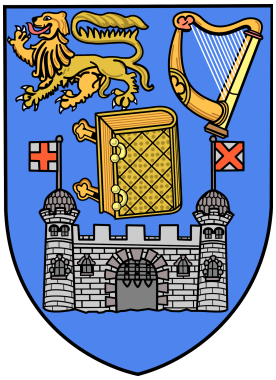
University of Dublin

TRINITY COLLEGE



A Multidimensional Visualisation System for a Relational Database of Students in a University

Submitted to the
University of Dublin, Trinity College
for the degree of
M.A.I. in Computer Engineering

Ferdia Soper Mac Cafraidh
Supervisor: Dr. Mike Brady

Submitted to the University of Dublin, Trinity College, May, 2017

DECLARATION

I, Ferdia Soper Mac Cafraidh, declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

_____ _____

Signature Date

Summary

This dissertation aimed to design a multidimensional visualisation system that would be capable of graphically displaying information maintained by a university on their students, stored in a relational database. This research investigated the graphics application part of the overall visualisation system architecture proposed for the implementation of a similar system. The research provides an overview of some of the existing literature on visualisation research, existing visualisation systems, and human factors in visualisation. This research is then applied to the prototype student visualisation system that was implemented using WebGL and the Three.js JavaScript library for WebGL, to create a browser-based visualisation system. The aim of this research was investigate the use of modern graphics libraries for the creation of multidimensional visualisations through the implementation of the student visualisation system and a review of existing visualisation literature and exploration of visualisation techniques.

The resulting application is a three-dimensional of a class of students, that is represented by the combination of multiple individual student cuboids. The students' information is encoded into their cuboid using a number of visual properties such as shape, size, colour, orientation and text. The visualisation is allows for users interaction through the use of the keyboard and mouse, to change opacity levels of student cuboids, rotate the class on all axes, and zoom towards and away from the class of students. This allows users to view the class from any angle or distance to allow for an exploration of the students that make up the class. The opacity level can be lessened for background students outside of a student of focus, which decreases the cognitive load of the task by removing distracting information. The cognitive load is also reduced using this feature, due to the opacity of the background students only being partially transparent, the user can still see the student in the context, thus removing the need for the user to remember how the student of focus relates, reducing the cognitive load due to a reduced memory required of the user.

The cuboid that represents each student represents information on student through the use of visual properties and text. The cuboid shows the student whose information is contained through the use of an identifying facial photo on the positive z-facing side of the cuboid, and through identifying personal details of the student on the negative z-facing side of the cuboid. The cuboid itself is composed of a number of separate year cubes that represent information on a student for a particular academic year. This time-dependent information includes the years in which the academic year occurred, the overall grade for that year of study, as well as the modules and relevant grades, including the timing of module per semester. This information was displayed on the x and y facing sides of the year cubes, which created a timeline for the data, which was shown to increase with negative changes along the z-axis. The academic years were made perceptually distinguishable due to the use of a colour coding for each academic year, and the length of a student's time relative to other students was clearly distinct by the difference in length of the cuboids, due to a different number of year cubes.

The result of the investigation found that modern graphics libraries allow for the relatively simple and flexible development of visualisation systems, by the implementation of the student information visualisation system. However the research also discovered that human factors, such as perception and cognition, have the potential for better use and the improvement of future visualisation design.

Acknowledgements

I would like to thank my supervisor, Mike Brady, for his continued help, guidance and expertise throughout the duration of the past year, which allowed me to complete this research and this tough academic year.

I would also like to thank my family and friends for their continued and unwavering support throughout the year and difficult moments, which made the completion of this work possible.

Table of Contents

1. Introduction

1.1 Aim and Background

The aim of this project was to research the effectiveness of a multidimensional visualisation, and to see how difficult it would be to implement using modern graphics libraries. To test this it was proposed that the visualisation was based on a student database that might be maintained by a university, motivated by the need for one in Trinity College to allow for easy access of student information. The hope was that the visualisation would provide a useful and meaningful way of finding student information, but the research conducted was also concerned with finding out why a multidimensional visualisation might be used instead of a normal tabular representation, and what additional benefits or drawbacks would there be with such a system.

The visualisation was developed using JavaScript and WebGL, using the Three.js library. The Three.js library allowed for the rapid development of graphics in WebGL. As a result of using JavaScript the page has to be hosted on a HTTP server, which shows that it is possible to do powerful visualisation work like this that can be easily distributed through a HTTP protocol as a service to clients, which would be the case if a similar system was implemented for the student records database in Trinity College.

1.2 Dissertation Overview

Chapter 2 - Literature Review: The dissertation begins with a detailed review of some major works of research in the areas that this project is concerned with. The section is broken down further to review each work individually to explain what the work covered and the views of the

authors on their work. The areas covered in this literature review involve the general areas of visualisation and the human factors involved in visualisation, as well as previous visualisation systems that have been developed.

Chapter 3 - Technologies: This section introduces the reader to the technologies that were used and that relate to the area of research. It introduces WebGL and Three.js, as these were the technologies used in the implementation.

Chapter 4 - Design: This section discusses the overall architecture of the student visualisation system that was designed, and then uses this show in context where the focus of research for this project fits in as a part of that system.

Chapter 5 - Implementation: The reader will then be introduced to the visualisation that was developed during the course of the research. The design and features of the system will be explained, only with details of how it was implemented. This section will also discuss issues and difficulties that occurred in the implementation.

Chapter 6 - Evaluation and Results: This section will look at the visualisation system that was produced by the end of the research, and closely examine its effectiveness. It will also look at what the system does well, and what the system fails to do.

Chapter 7 - Conclusion and Future Work: This section will look at what has been learnt by this research and what impact it could have in the real world. It will also look at the work that could be done to further the research from ideas that there was not time to implement, as well as from ideas gained by carrying out the research.

2. Literature Review

2.1 Designing the Visualization of Information

The work "*Designing the Visualization of Information"* [2] attempts to define precisely what it means to visualise information, as well as various methods with which this can be realised. The work tries to do this by defining clearly what is meant by visualisation, knowledge and information. This is then expanded upon to contrast the differences between knowledge visualisation and information visualisation. The paper states that a visualisation can be defined as a cognitive tool used to visually represent a set of data, with the aim of giving an insight to the observer. It is stated that information can be defined as the extraction of meaning from lower level data through a process, for example a statistical analysis of a set of data. This differs from knowledge, which they define as the result of a cognitive process, such as association or learning, which results in a better understanding of the topic area. The authors use these definitions of knowledge and information to explain that *Information Visualisation* is concerned with creating new insights from operations performed on abstract data, whereas *Knowledge Visualisation "aims to improve the transfer of knowledge between at least two persons or groups of persons"* [2, p. 6]. To summarise this, *Information Visualisation* generates an insight by exploring data through a computational process on the information, whereas *Knowledge Visualisation* is the communication of existing knowledge on a subject in a more efficient manner using graphics.

The paper continues to discuss a number of guidelines which could be of use during the development process of an information visualisation system. The first consideration is of the types of interactions that the user will need to perform when using the system. These will be a result of the type of user of the application, and will depend on whether the user is a data analyst or a consumer. Seven features are highlighted as typical requirements of systems that allow for

the effective searching of an information visualisation:

(1) Get an overview.

(2) Zoom in on specific items.

(3) Filter out items not interested in.

(4) Get details-on-demand.

(5) View how items relate to each other.

(6) Keep a history of user actions.

(7) Allow the user to extract sub-collections of information.

[2, p. 9]

Another key interaction that is highlighted by the authors is the *Focus-and-Context* or *Fish-Eye-View*, which can be seen as a combination of the interactions two and five in the list above. In the *Focus-and-Context* interaction the user is able to focus on a specific region of interest in the visualised information, while also being able to see how this relates to the rest of the information, thus providing context for the focused section.

The specific views of the visualisation that will be presented to the user are the next part of the design process that need to be considered. The authors highlight three questions that can be used to evaluate the designed views, and determine what views will be required of the system. These questions serve as initial steps in the design process to ensure that the needs of the end user will be met by the visualisation views that are selected.

1) *What can the user see?*
2) *What does the user need to see?*
3) *What would the user like to see?*

[2, p.9]

The authors continue the research by analysing the guidelines that have been proposed for visualisation design by previous research. They begin with the guidelines proposed by *Bihanic and Polacsek* [4, *"Models for Visualisation of Complex Information Systems"*], which state that the user profile should be used as an input to the visualisation. The authors say that this should allow for the restructuring of the visualisation of information based on the type of user and their needs.

The authors continue by discussing the work of *Eppler and Burkhard* [5, *"Visual representations in knowledge management: framework and cases*"], who propose a five question framework for the design process. These five questions, shown below, bear resemblance to the previously mentioned three design questions proposed by the authors of this paper. The authors state that *Eppler and Burkhard* [5] offer slightly different definitions of *Knowledge Visualisation* and *Information Visualisation* [2, p. 10]. They state that the differences lie in the extra questions which ask: what content needs to be visualised, what is the reason for the visualisation, and how can it be visualised. These differences can be observed in questions one, two, and five of *Eppler and Burkhard's* [5] design guidelines.

(1) What is to be visualised?

(2) Why is it being visualised?

(3) For who is it being visualised?

(4) In what context is the visualisation?

(5) How can the information be visualised?

[2, p. 10]

Forsell and Johansson [6, *"An heuristic set for evaluation in information visualization"*] propose a set of heuristics for a visualisation system, which the authors state can be considered as a set of guidelines in order to contrast them against their own work and previous works looked at by this paper. The authors state that the heuristics of *Forsell and Johansson* [6] evaluate a system's design, and also the user's role and interaction with the system [2, pp. 11-12]. They

state that the evaluation of the visualisation design needs to consider the method of data encoding and the effective use of space, with an aim to create a clean and organised view. Furthermore the design should also be consistent throughout the visualisation, and where possible the data set should be reduced to only essential information. This will ensure that the key information is conveyed by the visualisation, and that the user is not distracted by superfluous information. In relation to the user's role in the system, *Forsell and Johansson* [6] state the importance of reducing the cognitive load on the user, therefore the information should be presented in an intuitive manner to the user. The user should also only be required to perform minimal actions with the system to extract the information that they need. There should be a manner of flexibility to these actions, allowing a limited number of different ways of achieving the same goal when using the system. They also state that the functionality to provide additional information to the user must also be present in the system, for example through the use of prompting a user about the multiple methods of achieving a certain goal, which would allow them to choose the optimal method for themselves.

The authors continue with a discussion of *Elmqvist and Fekete's* guidelines [7, *"Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines"*], which the authors say deal with the addition of visual aggregates in an Information Visualisation solution.They say that aggregates should be used to visually summarise the data presented, and that they should kept simple to enable them to be easily interpreted. The authors highlight *Elmqvist and Fekete's* [7] other guidelines which discuss the importance of avoiding the overuse of aggregates, ensuring that the aggregates are easily distinguishable from the data visualised, and being aware of possible fidelity problems with the aggregates that can mislead observers about the data.

The authors also reference the work of *Keller and Tergan* [8, *"Visualizing Knowledge and Information: An Introduction"*], which they say describes the shortcomings of both *Knowledge Visualisation* solutions and *Information Visualisation* solutions, that need to be considered during the design process of a visualisation system. Difficulties with mapping concepts to visual representations and difficulties with available visualisation tools are stated by

the authors as the main shortcomings described by *Keller and Tergan* [8] when designing a *Knowledge Visualisation* solution. Whereas the shortcomings of *Information Visualisation* solutions are mentioned by the authors as technical problems when dealing with large data sets, and problems visualising information in a coherent manner.

The authors propose the use of a questionnaire for the design of an information visualisation system, following on the design principles and guidelines they have discovered during their research, and also based on the *Design Science Research Methodology (DSRM)* defined by *Peffers et al.* [9, *"A Design Science Research Methodology for Information Systems Research"*]. They authors state that the DSR methodology consists of the following six activities listed below.

1) *Identify the research problem.*
2) *Define the objectives of the desired solution.*
3) *Design and develop an artifact.*
4) *Demonstrate the use of the artifact to solve the problem.*
5) *Evaluate whether the solution artifact meet the objectives of the research.*
6) *Communicate the results of the research.*

[2, p. 15]

The authors' questionnaire would be presented to users of the visualisation who would evaluate the system by answering the questions, which would then be used as feedback for improving the visualisation system.

In summary the authors begin by defining what is meant by *Information Visualisation*, and the existing guidelines for designing these systems. They then use this information to propose a *Design Science Research* methodology that can be used to allow for the effective design of information visualisation systems. The authors mention that as this work only deals with the view stage of Information Visualisation, that future work could be done by expanding their design principles to cover the whole visualisation pipeline, starting from the transformation of low-level data to the visual representations which give the observers insight into the data.

2.2 Human Factors in Visualization Research

The paper begins by defining visualisation as a *"graphical representation of data or concepts"* [1] that are used as a cognitive support to assist humans with decision making. It states that visualisations use a number of different mechanisms for conveying information that utilise human perception to their advantage, and they also aim to make up for the cognitive deficiencies of humans. The authors also then make a brief contrast between two models of visualisation; a *continuous* model and a *discrete* model. These two models are, according to the authors, roughly related to the areas of *scientific visualisation* and *information visualisation* respectively. They state that *scientific visualisation* involves data with an inherently physical component, while *information visualisation* deals with abstract and nonspatial data.

The authors mention that over the last number of years, before the paper was written in 2004, that interest in *Human Factors* in the visualisation community has been increasing. They also say that user interaction in visualisation can be extremely valuable for analysis by means of exploration, and that interaction should not simply be seen as a *"means to the end of a good representation"*. However they state that these principles are sometimes ignored, and that despite growing interest in *Human Factors*, that in the *IEEE Transactions on Visualization Computer Graphics (TVCG)* journal that only 23 percent of TVCG papers included a human factors component in the history of the journal at the time of writing [1, p. 2]. The authors thus aim to help by providing a "*basis for studying human factors-based design in visualisation"* [1, p. 3] with a focus on *continuous model visualisation,* but also using a number of ideas and examples from *discrete model visualisation*.

The authors begin by briefly describing a number of *Human Factors* research approaches. The first approach is a *User Motivated Design*, which is the very basis of human factors research; research that is motivated by the end users. However they state that although this approach does consider the user to an extent, the main focus is on algorithm design rather than human factors elements, such as perception or cognition.

The next approach described is a *User and Task-Based Design*. This approach considers the tasks that a user would need and would want the visualisation system to be capable of performing. These tasks are identified by performing a task analysis, which may include interviews and surveys of potential users of the system. This process allows the designers to detail the specifications of the system and the potential limitations.

Perception and Cognition-Based Design is the next research approach described by the authors. These approaches use cognition and perception theories to create new and effective systems that take advantage of human capabilities and compensate for the human's shortcomings. The authors describe a basic set of guidelines based on cognition and perception theories which can be used when designing a visualisation system:

1) The visualisation should be domain and task specific, or if not it should consider the domain-independent subtasks which make up the domain dependent tasks of the user, such as zoom and overview.
2) Multiple data representations should be supported to allow for multiple types of users with differing tasks. It should be easy for users to switch between the various representations and continuity should be maintained, so as the observer is not confused, for example keeping the formatting consistent between views.
3) The user should always be able to see:
   - an overview of the entire data being visualised
   - the relationships between the data elements
   - the method of navigation and modification in the scene
   - the details of the current location data
   - the details of data in the local vicinity
   - their navigation history
4) Data that is the current focus of interaction should be displayed clearly and with the highest resolution possible.
5) User interactions with the visualisation should be maintained consistent throughout the system by reusing the same navigation tools.

Prototype Implementation is the next approach mentioned by the authors [1, p. 4]. They state that this approach involves rapid prototyping, which reduces the implementation time by allowing for more designs to be tested quickly, without having to invest a lot of time to constantly building complicated prototypes. In *Human Computer Interaction* (HCI) these prototypes could be carried out through the use of videos, storyboards, or software mockups. However the authors highlight that rapid prototyping has not been explored greatly for visualisation development. They mention that the development of graphics and visualisation libraries is improving the feasibility of rapid prototyping for visualisation, by reducing the time required by developers to create prototypes.

Human Factors play an important role in the testing of a visualisation system, according to the authors, and can be used to evaluate the functionality and ease of interaction of a system. The paper describes how this can be done through *user studies* and *usability inspections*. The authors state that user studies can provide both quantitative and qualitative data for evaluation of the system. Quantitative data could be obtained, for example, by measuring user accuracy in tasks, or by measuring the time taken to complete a task. Qualitative data could be obtained through the use of user surveys, interviews, or questionnaires after the test subjects have used the system. The advantages of using user studies for testing is due to their ability to quickly identify problems with the system. However it is also mentioned that there are certain disadvantages to this approach, which stem from the time consuming nature and complexity of designing the tests, and that they may not always identify problems or strengths of the system. The authors suggest that user testing could be better utilised with tests that evaluate specific design hypotheses, rather than general system tests.

Usability Inspections are also described by the authors as being a *Human Factors* method of testing a visualisation system. These are evaluation methods that have already been established in HCI for testing User Interfaces. The first method mentioned is a cognitive walkthrough of the system, which consists of an expert going through the system step by step, and attempting to think of potential problems that could arise. The next method mentioned by the authors would involve getting an expert to evaluate the system using a set of predefined

heuristics, of which the authors describe a number that could be used to test cognitive and multiple view aspects of the system [1, pp. 4-5]. However it is highlighted by the authors that it is unclear how well these approaches would work with visualisations, which generally have some ill-defined tasks, as the *Usability Inspection* approaches mentioned are designed for the testing of user interfaces, which have well defined interaction tasks. Another drawback mentioned of only performing usability inspections, is that by excluding end users from testing, it would reduce the ability to find unexpected errors in the system.

The paper summarises to say that each evaluation method could be used to find different problems with the system, and therefore to have an effective evaluation of a visualisation system that a variety of tests should be used throughout the design process. They suggest the use of *Usability Testing* firstly, to ensure that the design hypotheses were correct, and to identify interface problems that may affect *User Studies* results, and then *User Studies* could be used to evaluate visualisations, or to test specific hypotheses about the visualisation system.

The final *Human Factors* research approach described by the authors is *User-Centered Design*. They describe this as an iterative process which involves users as much as possible in each of the following four stages: task analysis, design, prototype implementation and testing. They state that development may begin at any stage during the cycle, but normally begins by analysing the tasks required, or testing of an similar existing system to discover its faults and limitations. The authors mention that this design philosophy is not widely used in visualisation, but they believe that it could lead to better visualisation design.

In the third section of this paper, *Tory and Möller* [1] introduce a number of examples of human factors methodologies and evaluations that exist currently in the visualisation research community. The first human factors method mentioned, is the use of perception in the research of visualisation designs. The authors discuss the use of preattentive processing which takes advantage of certain visual features, such as colour or length, which stand out to an observer, therefore allowing them to perform visual searches very quickly due to the perceived contrast. The authors discuss a paper by *Healey et al.* [10, *"Choosing Effective Colours for Data Visualization"*], which they say aims to improve the encoding of data in a glyph-based

multivariate display, through the use of preattentive features of the glyphs used. They state that *Healey et al.* [10] managed to encode a large amount of data in limited space utilising this method. It allowed for the preattentive processing of independent features, however the interest is generally in more than one feature at a time, which requires conjunctive searching. *Tory and Möller* [1] state that conjunctive searches are rarely preattentive, thus possibly limiting the utility of the multivariate data encoding by *Healey et al.* [10]. In relation to preattentive processing in visualisation they briefly mention two other pieces of research, one which developed a procedural method for developing preattentively distinct shapes, and another which looked at using visual textures for multivariate visualisation. The authors discuss how an interesting future study could compare these two pieces of research, to find out which approach is most valuable for preattentive processing and the encoding of multivariate data [1, pp. 5-6].

The authors continue their discussion of human perception by looking at research that uses the perception of colour to encode data. They state that visualisations frequently use a linear gradient of colour or intensity to encode information, such as topographic maps, which use colour to represent elevation. The authors highlight the importance of this research by stating that not all mathematically linear gradients are perceptually linear as well, for example the mathematically linear gray scale. They state that research in this area has focused on developing perceptually linear gradients of color value and saturation, and also in the identification of easily distinguishable colours which can be used to highlight parts of a visualisation to an observer [1, p. 6].

The paper then discusses the research that is being conducted within perception on how shapes are perceived in visualisations. They discuss that understanding the surfaces of irregularly shaped three-dimensional objects can be particularly difficult for observers, especially when the shape is semi-transparent, as is common with many medical applications of visualisation, in which they need to see overlapping regions. They highlight two pieces of research that have been conducted on improving the perception of three-dimensional shapes, one which used texturing, and another which used moving particles on the surface. They also mention how potential future work could compare these two pieces of research to determine when one is more appropriate to

use [1, pp 6-7]. The authors continue by discussing research into the use of geon diagrams as an alternative to Unified Modeling Language (UML) diagrams. They state that geon diagrams use 3D primitives (geons) to take advantage of human perception allowing for the remembering and distinguishing of shapes, which would replace the nodes and links used in UML diagrams, which they hypothesize would be easier for observers to understand. It is highlighted that a user study found lower error rates for relationship identification when using geo diagrams, compared to when using UML diagrams [1, p. 6]. The authors also discuss how research into shape perception is also being conducted on how *non-photorealistic rendering* (NPR) of 3D shapes can be used to improve the understanding of visualisations, compared to using photorealistic rendering. However they mention that while NPR styles for the improvement of perception may hold promise, that further work and evaluation is required in this area, before effective use can be made of NPR concepts in visualisations [1, p. 7].

The next human factors topic in research that the authors discuss is about the interactions between a user and a visualisation. They mention how research has been done on how to allow users to effectively interact with a three-dimensional visualisation. The problem stated by the authors is due to mapping movements of a movements of a mouse in a two-dimensional plane, to the movements of objects in three-dimensional space. Research has been conducted which found that manipulation of objects was easier when done relative to other objects than using absolute coordinates for manipulation. The authors also discuss research on the use of task-specific props for visualisation, such as the *"Cubic Mouse"* which allowed users to move along the three principal axes by manipulation perpendicular rods in a physical box [1, p. 7]. It is also discussed that research on this area is important due to the high amount of maneuvering required with visualisations when performing data analysis, between the visualisation and various tools and windows required for analysis. They mention a study that suggests minimising unnecessary navigation, which could be time consuming and cause a loss of focus in the end users, by integrating all the necessary tools required of an end user for a task.

The paper then discusses the human factors research that has occurred within Computer Graphics. It is discussed how research has been conducted to define mathematical models of

visual perception, of which the authors cite the *Daly Visual Differences Predictor* [11, *"The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity"*] and the *Sarnoff Visual Discrimination Model* [12, *"A Visual Discrimination Model for Imaging System Design and Evaluation"*] as two examples [1, p. 7]. The authors state how models of visual perception typically try to approximate factors of human vision, such as contrast, sensitivity and amplitude nonlinearity for the purpose of realistic image synthesis [1, pp. 7-8]. However it is stated by the authors that these sections of Computer Graphics and human factors are not particularly relevant to visualisation, where the goal is simply to represent data in a meaningful way [1, p. 8]. Thus more relevant research is from areas that look to improve rendering speed, which would allow for easier user interaction and better data analysis, and reducing image artifacts to reduce mistakes in interpretation and enhance perception. The authors discuss research which analised perception of images with an aim to increase rendering speed. They say one way that it was achieved by removing imperceptible details of the scene, which reduced the complexity and meant there was less to render, and in another work they achieved increased rendering speed by tracking the user's gaze, and adjust the resolution displayed accordingly, so as to reduce work being done on image artefacts without the user's focus [1, p.8].

The next human factor research area discussed by the authors deals with the use of human factors in the research of the development of effective transfer *Transfer Functions.* These transfer functions define how a voxel will be assigned a colour and transparency based on its intensity and/or spatial gradient value(s). The first method discussed in research for finding an effective transfer function is the use of *user exploration*. These methods leave definition of the transfer function up to the end user, however the authors state that for these systems to work that the user must either know what functions are useful beforehand, or be able to easily experiment with a range of different parameters. The authors state that this transfer function exploration by the users can be tedious and time consuming, thus needs an effective implementation for it to be a viable method of defining transfer functions in visualisations. The authors mention one study that attempted to implement this through the use of intuitive interactive widgets, and another which tried to provide extra information on the data such as histograms to provide more information to the user and aid them in choosing an effective transfer function. However they

also mention that neither of these works conducted user studies to evaluate and verify their hypotheses. They continue to describe that a promising way in which to use user-defined transfer functions is through the use of a history tool, which would show the users what transfer functions they have already used. They mention two studies that tried to achieve this, one of which used a graph to show the relationship between transfer functions with corresponding images for the effect, and another that used a spreadsheet with corresponding images [1, p. 8].

The authors continue to discuss research on parameter selection for transfer functions. They state that there is great difficulty in choosing what parameters to use, and that in general more parameters will increase the flexibility of a transfer function, but adds complexity in finding a good combination of parameters as a result of the increased search space. It is stated that many visualisations will simply use intensity values alone to define the transfer function, while others will also combine the first and second derivatives, which are the spatial gradient values and boundary areas. They mention a study conducted that developed a visualisation in which users were allowed to specify specific spatial regions of interest to have increased opacity [1, pp. 8-9].

The authors mention further transfer function research that has been conducted on trying to generate transfer functions automatically and semi-automatically. The automatic functions would generate the transfer functions with little or no user input. The authors state these processes cannot utilise user intuition to their advantage as a result, and also that it could be harmful to leave users out of the process, as this could reduce the ability for data exploration by the user and limit their understanding of the visualisation images. This is what the semi-automatic transfer function generators try to avoid by introducing limited user control into the process, such as the work by *Kindlmann and Durkin* [13, "*Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering"*], whom the authors say constrained transfer functions to those that display boundaries and allow users to modify how boundaries are emphasised by the transfer function [1**,** p. 9]. This made it easier on the users due to the limited parameter space, while still allowing them to utilise the transfer function for exploration of the data set.

The authors also mention research done on using visual search to find good transfer functions, turning the user's search from an abstract mathematical search to a visual one [1, p. 9]. One study they mention attempted this by generating many transfer functions and sample images and allowed users to select the useful ones, while another study implemented it by directing future visual searches based off of images chosen by the user that were close to the desired image. The authors state that these strategies are heavily dependent on *"human visual search strategies, and require substantial user testing that has not yet been done"* [1, p. 9]. However they reiterate that, similar to fully automatic methods of generating transfer functions, it could be counterproductive to remove the end user from the process.

The final method of defining transfer functions discussed by the authors attempts to aid users in defining them by restricting the input parameters possible based on rules about the data structure, the visualisation goals and perceptual knowledge, so as to aid the users to select good transfer functions. They give examples of visualisations where the user's goals are matched against finding an appropriate coloring scales for the task at hand, such as suggesting using perceptually linear colour gradients for viewing medical images where the goal is to gain an impression of the overall structure. This technique of defining the transfer function aims to avoid useless combinations of parameters, by restricting the user from entering them together.

The authors move on to discuss the impact on human factors research in determining how visualisations balance *detail and context* of the information displayed. Many visualisations will deal with vast amounts of data, and the more data displayed in a visualisation the less detail that will be shown about each individual item in the visualisation. This can be overcome by reducing the amount of data displayed concurrently to include more detail on each item, however the authors state that this can lead to less awareness of the information in context of the entire data set, and puts extra cognitive load on end users to retain the context information mentally. The authors introduce how this topic is being researched with regards to two-dimensional graphics, before continuing on to how it is being dealt with in three-dimensional displays. The authors begin by mentioning the screen real-estate problem in relation to two-dimensional displays, which is the problem of fitting large amounts of information into a small space. They mention

the use of *detail and overview* techniques which can use multiple windows, one of which displays the detailed local information, and another which shows the information in a global context. However they state that these extra displays could cause a cognitive overhead. The alternative technique to this mentioned by the authors is *focus and context*. In this technique the authors say that a single display is used to reduce cognitive overhead, and use image distortion or relative size distortion to allow users to focus in on a section of the data in detail, while still seeing it in a global context of the distorted or relatively smaller background.

The authors continue to discuss how the screen real-estate problem is equally prevalent with three-dimensional graphics being projected onto two-dimensional displays, but further complicated with the extra complication of object occlusion. They state that attempts to solve this problem have been done in one case using an *overview and detail* method similar to the idea in two-dimensional graphics already mentioned, and also through the use of distortion techniques such as using a fisheye lens which has been done several times by different research groups [1, p. 10]. It is mentioned by the authors that these distortion techniques are most effective when the data has a regular shape, or through the use of a three-dimensional grid which makes it much easier for users to understand the distortion by seeing the bends in the grid lines. However the authors state that distortion methods are harder to implement with volume data sets which can frequently occlude any 3D grids drawn for reference. They mention that not much research has been done in the area of using *detail and context* displays with volume data, and highlight that user interaction with distortion lenses may aid to increase the understanding of how the distortion works, which could be evaluated through a future user study.

A number of methods of attempting to avoid the occlusion problem in *details and context* displays for 3D Graphics are then discussed. They describe a number of ways in which this problems has tried to be solved, through the use of clipping planes, which only render the objects between the clipping planes, and transfer functions used in direct volume rendering which made the voxels semitransparent. The problems with clipping planes was removing the focus area from context as data outside the clipping plane is not shown, conversely the transfer functions showed all the information with better visibility of regions of interest (ROI) due to varying opacity

levels, it does not increase their allotted screen space, meaning less detail is shown. They mention other research which attempted solutions to the *detail and context* occlusion problem by allowing the users to view the visualisation from multiple angles concurrently and using different transfer for different perspectives, while another tried to allow for different rendering styles in ROI of the data. However they state that again like the transfer functions, these solutions do not increase magnification of ROI, and while showing promise, the work is still in its infancy at the time of the paper.

The paper continues on to the subject of Human and Computer Cooperation in the context of visualisation systems. It is stated that most visualisation systems are designed for this cooperation, and in particular allowing for the computers to take charge of storing and displaying the data, while the humans role is to interpret and understand the data. They state that the computers involved could be better utilised to aid users in processing their own mental model of the data while trying to understand the visualisation, through providing an analysis history and platform for users to visually describe their understanding of the data. This could use cognitive and perceptual models of human factors to improve the interaction, and also possibly aid the users by recognising errors or limitations of the user's understanding of the visualisation. The authors cite a paper that says that current computer-based tools at the time did not do enough support this *"integration of insight"* [1, p. 11]. The authors mention that despite these many tasks which visualisation systems could perform and could provide a valuable addition to data analysis, that current research at the time of writing is almost entirely devoted to methods of visually representing the data to enhance data analysis.

The authors finish by stating that while recent work in cognition and perception-based visualisation design has produced many new ideas, that cognition and perception are not being used to their full potential, therefore there is still a lot of promise into further research in this area within visualisation design. They also list a number of potential directions for future work in this area, shown in the list below:

- *Determining when, if, and how increasing display size and resolution (independently and together) affects performance at visualization tasks.*

- *Empirically comparing techniques (detail and context methods, transfer function specification methods, multivariate data visualization techniques, shape enhancement methods, NPR methods, etc.) to determine when each method is more suitable over other comparable methods.*
- *Performing user studies to consider whether histograms (and similar data) support transfer function specification and which data is the most useful.*
- *Developing and evaluating task-specific input devices to aid interaction.*
- *Reducing unnecessary navigation within and between tools, through better display design, and the integration of tools based on task requirements.*
- *Developing tools that provide cognitive support for generating insight and for the organization of ideas.*
- *Exploiting perception and cognition theories that have not yet been considered in visualisation design.*

[1, p. 11]

The authors highlight that many parts of human factors-based design have not yet been explored in depth by visualisation research, and also stating that current research procedures are "*ad hoc and informal"* [1, p. 11]. They state that researchers rarely perform iterative user-centered designs, or perform a structured task analysis, and that rapid prototyping is still not widely used in research. However they say that these approaches could speed up research progress by decreasing the amount of time and effort wasted on ineffective designs. It is also highlighted that due to a lack of stringent evaluation procedures in many visualisation research projects, that the authors have difficulty determining which particular pieces of research show promise. They conclude that the visualisation research community must develop their own specific methodologies, but that this can only occur once current HCI methods have been adopted and evaluated, which will allow researchers to determine what needs to be changed to suit visualisation research and implementation [1, p. 12].

2.3 Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases

The authors state that Polaris was created as a system for to aid in the exploration of multidimensional databases through the use of rich graphical displays. The creation of Polaris is shown by the authors as a possible solution to the need to extract information through analysis from the increasingly large databases being used in a variety of applications. Polaris aims to allow for an exploratory analysis of data, which the authors describe as being unpredictable and as a result requires for the easy manipulation of the presentation of the data so as the analyst can compare their hypotheses and discover information easily. Polaris aims to extend the standard PivotTable interface to allow for richer graphics which will aid this exploratory analysis of the data.

The authors state that Polaris has been designed for interactive exploration with large multidimensional relational databases, which organise data into many interrelated tables of heterogenous data. They also state that each row or tuple in a table describes an entity, and that each column with that row will define a single property of that entity. Polaris categorises the data stored as either quantitative or ordinal, treating interval data and nominal data as quantitative and ordinal respectively to reduce the number of potential data types from four to just two. The authors also state that the various fields within a relational table can be classed as dimensions or measures, stating that they are similar to independent and dependent variables in a traditional analysis approach [3, p. 2]. They state that Polaris treats all ordinal fields as dimensions, and treats all quantitative fields as measures.

The authors describe that in many databases an entity will be described by multiple dimensions, and that in most multidimensional databases they structure the data as n-dimensional data cubes, with each dimension of the cube relating to one dimension defined in the schema of the database [3, p. 1]. It is stated for an analysis tool for large multidimensional databases to be effective that it must support data-dense displays containing a large number of records and dimensions to visualise large subsets of the date, multiple display types which are suited to the different tasks involved in data analysis, and also an exploratory interface that allows for rapid

changing of both the data being viewed, and the view of the data due. The authors say they have addressed these demands in Polaris by giving the user a platform for the rapid and incremental generation of table-based displays of columns, rows and layers. They state that tables are particularly effective for displaying multidimensional data because they can represent the multiple dimensions through explicitly encoding the structure into the table, they also allow for easy comparison of data across dimensions, and they provide a familiar interface as tabular data has an extensive history of use [3, p. 2]. The authors state that a *visual specification* is defined by the configuration of fields in the axis shelves and layers when dragging and dropping information. This visual specification is used by Polaris to generate graphics, which the authors break down into three components: *Table Algebra*, *Types of Graphics*, and *Visual Mappings*.

The *Table Algebra* is implicitly created through the dragging and dropping process of data onto the axes and layers, by the position of the data and its type, whether it is ordinal or quantitative. The authors state that "*ordinal fields will partition the table into rows and columns, whereas quantitative fields will be spatially encoded as axes within the panes"* [3, p. 4]. The authors define the semantics of each of their three algebraic operators, which are the *Concatenation, Cross,* and *Nest* operators for combining sets. The *Types of Graphics* are specified by the data of the axes in the configuration of the table specified by the user, which the authors state is split into three types of *ordinal-ordinal*, *ordinal-quantitative*, and *quantitative-quantitative*, which are further defined by the number of independent and dependent variables as well. Then the user is able to choose the type of mark they want to represent the data out of the various marks available within each of the three types [3, pp. 7-8]. The mapping also needs to determine the precise form of the data transformations, looking at how the data is grouped and whether aggregates are formed to choose suitable graphic types.

The final stage of the visual mapping is the encoding of the fields of records to visual or retinal properties. Polaris uses visual properties based on the the retinal variables specified by *Jaques Bertin* [14, "*Semiology of Graphics"*], which the authors say are shape, size, orientation, colour and texture [3, p. 7]. All of these variables are available for data encoding apart from texture in the current version of Polaris at the time of writing [3, p. 7]. The authors state that by allowing the users to explicitly encode different data to retinal variables it allows for greater data

density in the visualisation and for a greater variety of displays as well. However they state that in order to maintain a concise specification for the encoding of the data, the user should only specify the visual property to be used in the encoding, and then the system should take charge of mapping from the domain of the data to the range of the visual property [3, p. 7].

The authors of the paper continue to say that for the interactive exploration of data, that additional interactivity is needed extra to the interactive features already outlined, which included rapid table configuration, and selection of graphics and visual encodings for the data. They state that the resulting visualisation must be manipulable and they describe four ways in which Polaris achieves this: deriving additional fields, sorting and filtering, brushing and tooltips, and undo and redo [3, pp. 8-9]. The authors describe that the derivation of additional fields is necessary because it allows the author to visually describe the discovered information and relationships of the data. It also allows users to reduce the size of large data sets by hiding uninteresting data, which allows for an easier understanding and analysis of the information [3, p. 8]. They state that in Polaris the additional fields are derived through aggregation, such as summation, average and thresholding of values, and through the partitioning of quantitative data or ad hoc grouping of ordinal data. Polaris also allows for manipulation through the sorting and filtering of data, which is important as often only with the sorted crucial information can insights into the patterns and trends of the data be found. Filtering out of unimportant information allows more room on the screen for areas of interest, and also allows for greater user attention on these areas. Sorting allows the user to group together interesting values to find trends. Polaris allows for users to sort through *dragging-and-dropping* of values, rows and columns, and also through programmatic sorting where one field can be sorted based on the value of another field [3, pp. 8-9]. The third method of manipulating the visualisation provided by Polaris is through brushing and tooltips. Brushing allows the user to select data in one display by drawing a rubberband around a selection, and this selection of data is then highlighted in other displays, which can show the user relationships and correlation of the data and displays. The tooltips gives the user details on demand by hovering over a data point or pane in the visualisation. This extra information given can allow users to better understand the relationship between the graphical marks and the underlying data [3, p. 9]. The final interaction provided by Polaris is through undo

and redo actions when manipulating the visualisation. It is extremely important as it allows the user to quickly change the path of exploration of the data, and also promotes experimentation with the visualisation as there is no danger of losing progress. Polaris manages this process by saving the visual specification at each stage, and then the undo or redo updates the display to the relevant saved visual specification [3, p. 9].

The paper continues to describe how the visual specification that is defined, aside from defining the appearance of the visualisation, it will also define the SQL queries to the database to gather the relevant information to build the visualisation. It is stated that this is done in three steps. The first stage defines a query, that will return all values for a given field, or subset of a field, to filter out unwanted information. This would generate a query as shown below, with the *filters* specifying the relevant filter.

```
SELECT *
WHERE {filters}.
```

Fig. 2.1 [3, p. 10]

The second stage partitions the data into groups that will correspond to the panes of the table defined by the user. This query must iterate over the table and repeatedly execute the following query to generate each pane, as there is no way of partitioning within a single query according to the authors.

```
SELECT *
WHERE {Row(i) and Column(j) and
       Layer(k)}.
```

Fig. 2.2 [3, p. 10]

The authors state that the third and final stage of this process deals with the transformation of the records that are in each pane. This stage creates aggregates for each field by using the *SUM* aggregation by default, or the user specified aggregation operation if specified. Aggregate filters may now be applied in this stage, as they could not be previously applied according to the

authors. The query in this stage will group the aggregates, apply the filters and order the fields by name in the sorting shelf, however if no aggregates are mentioned in the visual specification, the resulting query will simply order the records by their names in the sorting shelf [3, pp. 10-11].

The authors then outline two scenarios to show the effectiveness of Polaris as a system for the analysis of multidimensional databases. They state that by formally categorising the graphic types, they are able to provide the users with rapidly changing visually displays of the data. These displays allow the users to test their hypotheses about the data through exploration of the data using the easily customisable visualisations of the data to show trends and relationships of the data [3, pp. 11-12].

The authors begin the discussion by contrasting the Polaris visualisation system, with the visualisation system of Wilkinson [15, "*The Grammar of Graphics"*]. They state that the Wilkinson's visualisation system differs from Polaris due to the data models used in each system. In Polaris they used a relational model which they say is a major advantage because it leverages existing multidimensional relational database technology to its advantage, meaning visual specifications can be easily interpreted as SQL queries [3, p. 12]. The authors state that Wilkinson intentionally does not use a relational data model as he states there are shortcomings with the support of the relational model in regard to statistical analysis tasks [3, p. 12]. Apart from this advantage, the authors state that it is hard to tell which visualisation system is superior.

The authors continue on to discuss the performance of Polaris in interactions. They state that their research focused on defining techniques, semantics and formalism to generate visualisations, rather than focusing on the performance time of the queries generated by the visual specification to gather the data from the database. They say however that they found no negative impact from large query response times for exploration of the data, even with response times of tens of seconds. It is mentioned that due to the small number of tuples returned when using the Pivot Table tools, and as a result server-side query time is the main constraint on performance, rather than manipulation and drawing of the information on the client-side [3, p. 13]. As a result the authors state that they would like future work to include further testing with large data sets, having already tested one set containing 6 million tuples, so as to focus on the performance of queries and performance of the Polaris system through other means [3, p. 13].

The authors mention their future plans for work on the Polaris system which include prefetching data and caching strategies to achieve real-time interactivity, generating database tables from a selected set of graphical marks, and extending the shelves configuration from x, y and layer, to also include an animation shelf. They say that the animation shelf would allow users to see an animation of one field interacting with another, for example using the Month field would allow user to see how data changed over time [3, p. 13]. It is stated that the hope for the table generation from graphical marks would allow for the development of lenses that would be capable of much more complicated data transformations, as the transformations would be performed in image space rather than data space.

Overall the authors conclude that Polaris provided *"an interface for the exploration and analysis of multidimensional databases"* [3, p. 13], thus it achieved its goal through the extension of the classical Pivot Table interface. They state the other major achievement from this research was the concise visual specification that they defined for describing table-based graphical displays of relational data, and the interpretation of this visual specification into relational database operations.

3. Technologies

3.1 WebGL

WebGL is a low-level API for rendering 2D and 3D graphics in any compatible web browser without the need of additional plug-ins [17]. It is based on OpenGL for Embedded Systems (*OpenGL ES*) and uses JavaScript code combined, run on the CPU, with shader code that is written in OpenGL Shading Language (GLSL), which is a C-like language that executes code on the computer's GPU. WebGL uses a right-handed coordinate system as demonstrated in the image below, which illustrates the difference between a left-handed and right-handed coordinate system. The difference between the two coordinate systems is on the positive and negative directions of the z-axis, as in a left-handed system the z-axis is positive going away from the screen, but in a right-handed system it is positive towards the screen, as it is in WebGL.

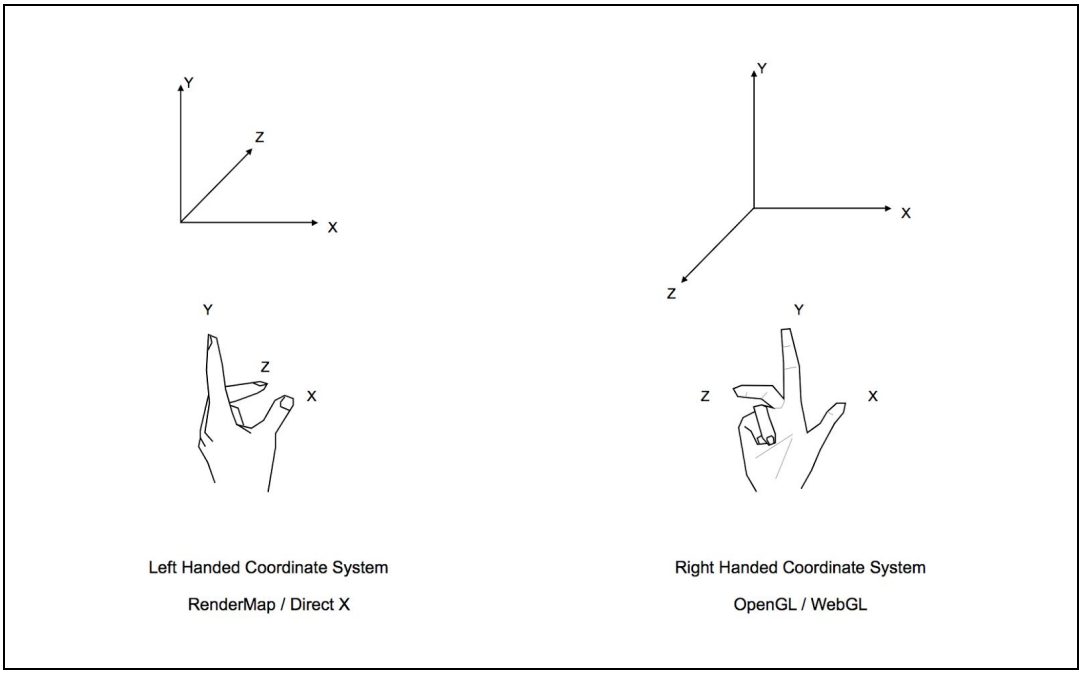


Fig. 3.1 [16]

The current version of WebGL is WebGL 2.0 which is based on OpenGL ES 3.0 [17], while the original release version, WebGl 1.0 was based on OpenGL ES 2.0. WebGL stays very close to the specifications of OpenGL ES, with slight differences due to the automatic memory management capabilities expected in using JavaScript. This effective combination of these technologies allows OpenGL ES users to transfer their skills to designing easily portable web applications that are possible using WebGL, and while also conforming to the standards of JavaScript which is one of the most widely used programming languages and a corner stone technology of web development.

WebGL works with most modern browsers and many major "browser vendors Apple (Safari), Google (Chrome), Microsoft (Edge), and Mozilla (Firefox) are members of the WebGL Working Group" [17]. It is also possible for WebGl to be run on many mobile devices through their browser such as on Android using the Chrome Browser. WebGL uses the HTML canvas element as its destination for rendering in a HTML webpage.

WebGL was designed by the non-profit Khronos Group with the initial WebGL 1.0 specification in March 2011 [17]. They maintain WebGL and it is one of their many working groups as part of their goal for developing royalty-free graphics application programming interfaces.

3.2 Three.js

Three.js is a JavaScript library that was developed on top of WebGL for the rendering of 2D and 3D graphics [18]. It is a high-level library that greatly reduces the complexity of WebGL, to allow for the easy creation of complex graphics without the requirement for additional browser plugins. The library is contained within a single JavaScript file that a user can either link directly from the HTML page to the repository on Github [19], or download a local version and use a http-server to run it locally from their own machine. Three.js is available under the MIT licence which allows for both private and commercial use, distribution and modification of the Three.js API [19].

Three.js contains many features that speed up the development process for creating graphics and animations. It contains premade geometries for three-dimensional shapes, such as cubes and spheres, as well as many facilities that greatly ease the process of loading images that can be used for texture mapping. It also provides functionalities for interaction with the scene, through a library which allows you to add an orbit controller to a scene which can then use the house to rotate and zoom into or out of the scene. The following is a general but not complete list of the areas that Three.js provides functionality for end users:

1) Animation
2) Audio
3) Cameras
4) Lighting
5) Geometry
6) Raycasting
7) Helpers

8) Loaders
9) Texturing (Including predefined materials)
10) Maths Libraries (Quaternions, Vectors, Matrices, Linear Algebra)
11) Scene Fog

[18]

Three.js is one of the most widely used graphics libraries in conjunction with WebGL and as a result there are a large number of previous works on their website which show the possibilities of what can be implemented using this library. An extensive list of applications that were implemented using Three.js can be found on their website [18], to demonstrate what is possible through this library.

```
// create the camera
camera = new THREE.Camera( 70, window.innerWidth / window.innerHeight, 1, 1000 );
camera.position.y = 150;
camera.position.z = 350;
camera.target.position.y = 150;

// create the Scene
scene = new THREE.Scene();

// create the Cube
cube = new THREE.Mesh( new THREE.CubeGeometry( 200, 200, 200 ), new THREE.MeshNormalMaterial(
cube.position.y = 150;

// add the object to the scene
scene.addObject( cube );
```

Fig. 3.2. *This image shows code using the Three.js library to create a scene with a camera pointed at a cube*. [20]

3.3 JavaScript

JavaScript or simply JS is a high-level object-oriented programming language and it is one of the three core technologies that is used in web development, along with HTML and CSS. JavaScript

is used to create dynamic web pages that can react and change based on the actions of the user, or the state of the machine accessing the web page. The language is supported by all modern web browsers [21].

JavaScript is a scripting language. This means that the language itself is not compiled before runtime, but it is interpreted at runtime and evaluated. One of the useful features is the dynamic typing of variables, as is similar with other scripting languages such as Python. This means that each variable is not bound to a specific type, such as an integer or string, for the entirety of the script. For example a variable that was bound as a string, can later be rebound as a number, or vice versa [21].

Javascript can be embedded directly into the HTML pages themselves through the use of the *<script>* tag, or it can be linked via an external file. As a result the code written in JS is executed on the client side, compared to other scripting languages which execute on the server side, such as PHP. This gives the advantage of providing more interactability with the web page and being able to respond to user interactions immediately, as for the action to be evaluated and a response generated, it does not have to send a HTTP request to the server and receive a response. Javascript has great flexibility and support in the number of libraries which can expanded and facilitate development, such as JQuery which is one of the most well known JavaScript libraries [21].

There are some disadvantages of JavaScript due to the fact it is executed on the client side, and not on the server side. The main disadvantage is that end users are able to turn off scripts from running on their machine. If this is done, this could greatly affect the performance and usability of a page using JS to display certain elements of the page [21].

4. Design

4.1 System Architecture

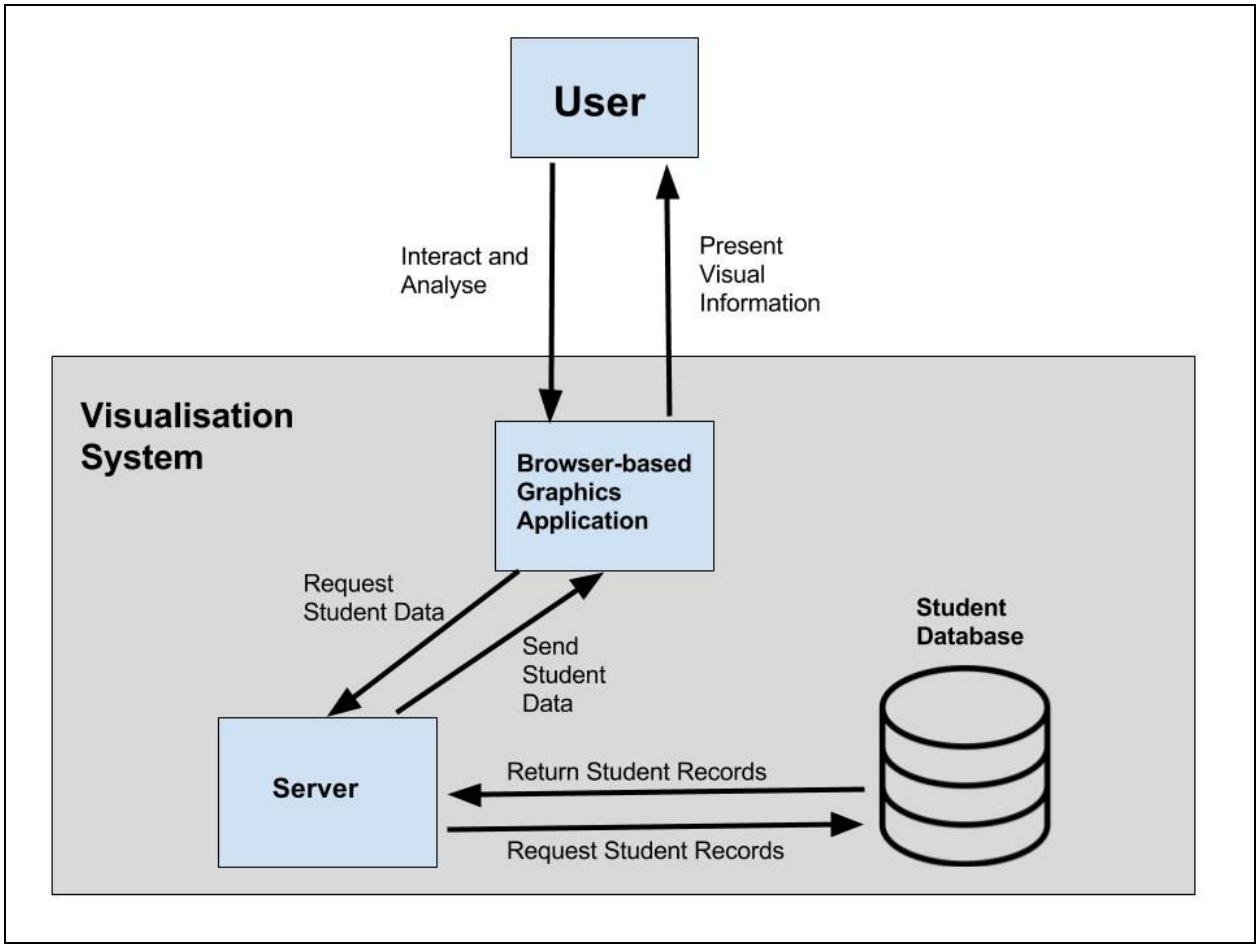


Fig. 4.1: *Diagram of the Visualisation System Architecture*

The overall visualisation system was designed to be a system that could be implemented in a university setting, and would allow for the visualisation and insight into students' information that may be stored by the university. With this in mind there were a number of design goals in mind when thinking of the overall architecture of the system. The system would be used by a

wide variety of members of staff in the university, so the system would need to be easily accessible, as well as being easy to manage updates and initial installations on user machines. Due to these needs, the visualisation system was designed as a web service that users would access through a browser. Designing the visualisation system as a web service meant that users would not need to install additional software to their machines to access the service, and that future updates would also be easily be deployed to the system without having to upgrade individual user machines as well.

The visualisation was designed with three main components, that could be grouped into frontend and backend development. The first component would be the work done in the browser, which would be the frontend development work. This would be the graphics application that would transform the data to visual representations, and also provide a means of interaction between the user and the visualisation. The backend development would contain the other two components of the system design, which would be a server and a relational database. The server would have two tasks which would be serving information to the browser application using a HTTP protocol, and also interacting with the relational database to extract information on particular records. The relational database would be used to maintain all of the information available about individual students in the university.

There would be a number of tasks to implement this system architecture, broken down into the three components of the system. The relation database that maintains the student records would need to be designed. This would involve designing a relational schema for the tables in the database and the keys that would link the tables to each other. The next component would involve the implementation of the server. It would need to be capable of interacting with the user's browser via a HTTP protocol, for the transfer of information on the students as well as the HTML page which runs the application. The server would also need to be capable of translating the user's requests for information into SQL queries that could retrieve the information from the database. There likely could be a large amount of development time devoted to this task of translating user interactions into SQL queries, as seen in the work done developing the *Polaris*

system [3], albeit the work would be simpler in this case as it would be a specific application and not a general visualisation tool like *Polaris* [3]. The third component of implementation would involve the graphics application, which would be a large section of work and need to use an effective means of transforming the data into visual properties, as well as handling the interactions that the user would need to perform exploratory analysis of the students' information.

4.2 Research Focus

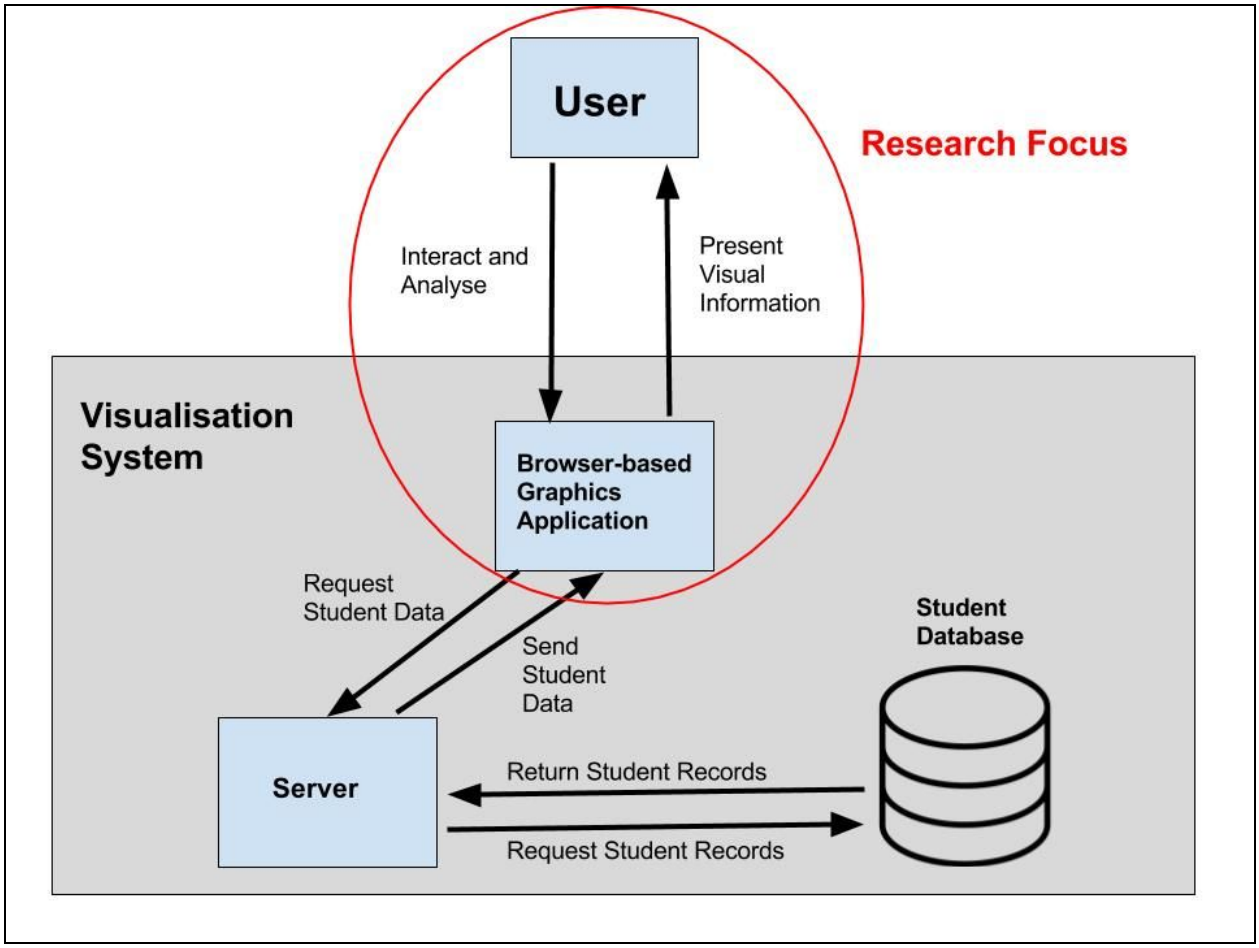


Fig. 4.2: *This image shows the research focus was on the browser-based graphics application.*

As discussed in the previous section, the complete visualisation designed would be quite a large task to implement, containing three different components with separate tasks. Due to the limited time during the course of this research, it was decided that the implementation would focus on the development of a prototype of the browser-based graphics application. This meant that research would be focused on the current graphics libraries that can be used with modern browsers, to select a suitable library that would be flexible but also allow for rapid development and experiment. The chosen technology was the Three.js library which is a JavaScript API for WebGL. This was selected for its' simplicity allowing for greater speed in development, its' compatibility with most modern browsers, and also due to the familiarity of JavaScript and OpenGL which were possessed prior to the research which would allow for easier development using these technologies. The research focuses on using Three.js to investigate a number of data transformations for converting the student information into visual representations. These visual representations aim to use human cognition and perception to make the information more accessible than textual data, allowing the users to gain insight into the student information. The research also focuses on the investigation of intuitive techniques for allowing the user to interact with the visualisation.

5. Implementation

5.1 Database Assumption

The visualisation system was designed to be used with information being pulled from what was presumed to be a relational database that would be maintained by a university. However as no such data source was available for use, the system uses randomised mock data at runtime. The system was designed with the intention of being converted to an actual data source, and is thus quite flexible for changing variables on how to render students information or how many students are contained within the data source.

As mock data was used in this implementation of the visualisation design, a number of assumptions were made in assuming what data would be stored by a university on its students. It was assumed that the information would contain an academic transcript for the student, detailing their grades for each module and year that they had been in the university, as well as knowing which course the student was on and which classes they were currently taking. The data was also presumed to contain some personal information on the students, which would contain an identifying photo of the student and information such as their name, age, gender and current address, etc. This is the total of the information that was to be visualised by the system for the user.

5.2 Data Transformation

The aim of the data representation was to try and encode as much information as possible implicitly into visual properties of the view, such as colour and size, and in other cases to use textual information in a concise and clear manner. The colours, text and pictures were added to

the student cuboids using UV mapping, which maps a two-dimensional image onto a three-dimensional surface.

```
//create group of year cubes for student
function makeStudent(student_id, num_of_years, x, y, z) {
    //setup cubes and constants
    var year_group = new THREE.Object3D();       //create an empty container

    for (i = 0; i < num_of_years; i++) {
        var needs_face = false;
        if (i ==0) {    //only first cube needs face
            needs_face = true;
        }

        //create the year cube
        var year = makeCube(cube_dim, cube_dim, cube_dim, new THREE.Color(year_colours[i]), needs_face, student_id, i+1);

        //translate it back
        year.position.set(0,0, -(i*cube_dim));

        //add it to the student group of years
        year_group.add(year);
    }

    year_group.position.set(x, y, z);

    return year_group;
}
```

Fig. 5.1: *This image shows the function used to create student cuboid as a Three.js Object3D type.*

5.2.1 Shape

In the visualisation, each student is composed of a number of blocks or cubes. These cubes are all of uniform size, and combine together to form a cuboid which contains all of the information on a particular student. The uniform size is maintained for each cube year so as to allow for easy comparison and reduced cognitive load during analysis by the end user who may want to contrast different year cubes.

The visualisation groups these student cuboids together to form a large cube frame shape that represents each of the students from a given class, or other grouping which may be possible in the visualisation. A class grouping may contain a number of different courses, which have students in different years taking the same class. This year difference is easily distinguishable between student cuboids due to the variations of length, due to the number of year cubes they contain. This distance was chosen to be easily perceptible, thus allowing for the easy visual analysis of the year groups that make up a particular class in a university.

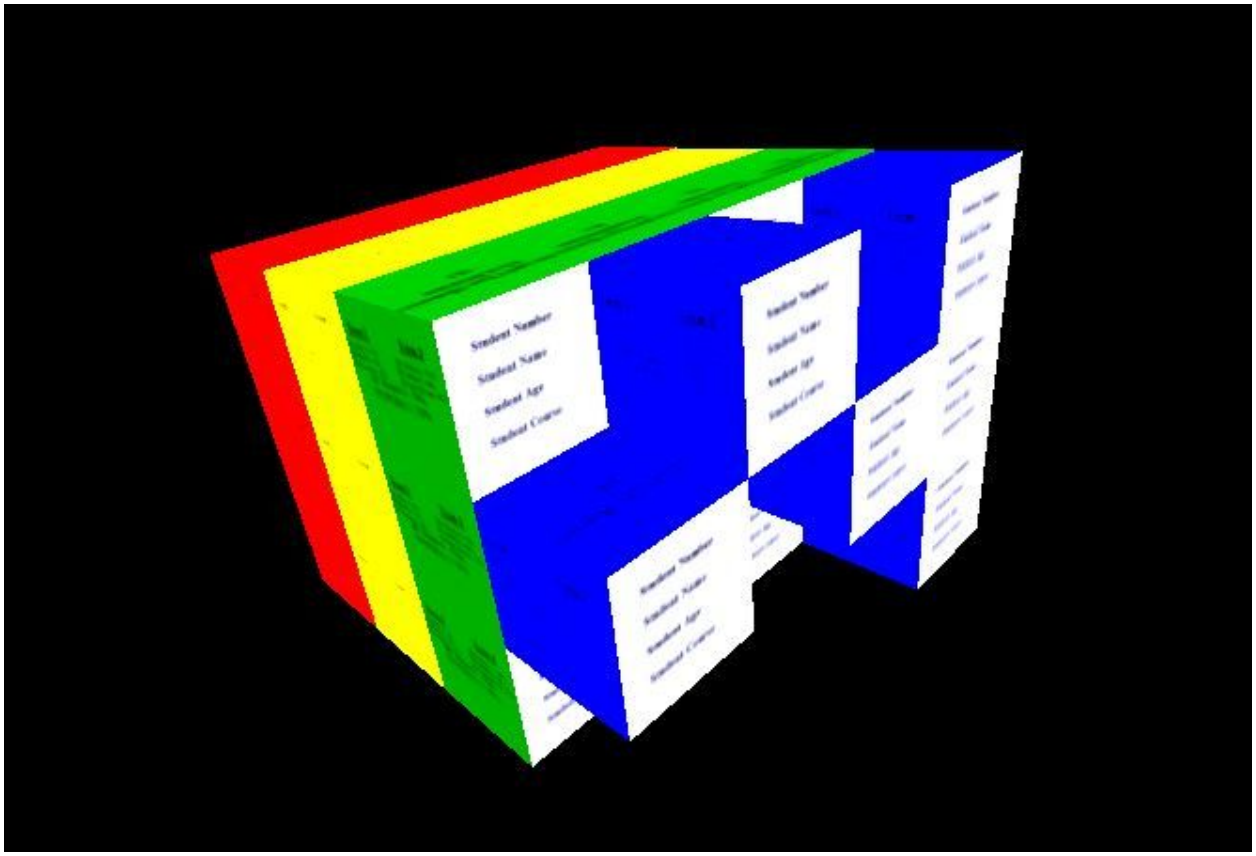


Fig 5.2: *Illustration of the visible difference in cuboid length.*

5.2.2 Colour

Each cube of the student cuboid represents one of the years in which the student has been enrolled in the university and there exists information on the student. These year cubes are given

specific and distinct colour codes so as it is perceptually easy to distinguish the year cubes. The colour coding remains the same for each student, so there is a direct correlation between a colour, and the year that student has completed. This allows for a quick and simple visual search to see how long a student has been in the university, and also allows the user to note that the information on that cube relates to a specific academic. This means it would be easy to identify students that had repeated a year, as not only would they have a longer cuboid than other students, representing a longer time in the university, but they would also have longer sections of the same colour, indicating that a particular year had been repeated. This colour coding of the academic years allows for a highlighting of potential regions of interest to the user analysing the student information.

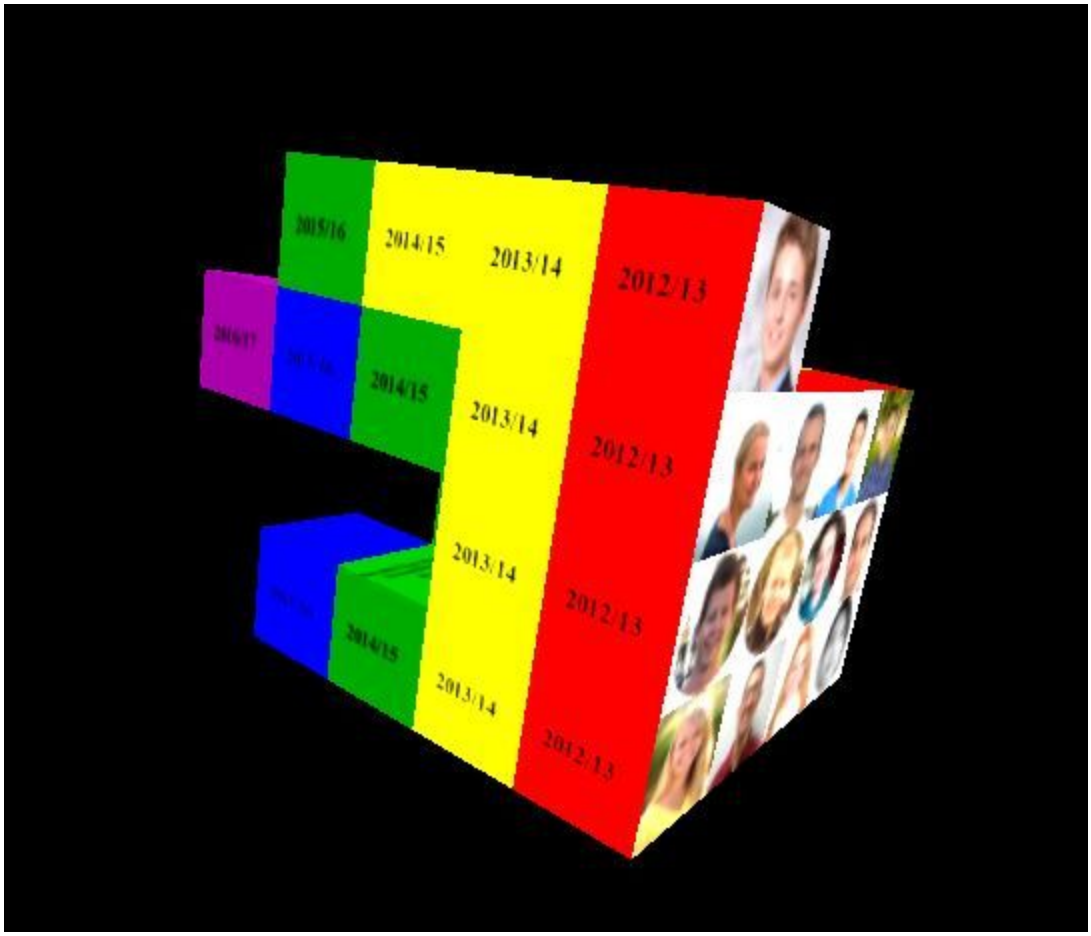


Fig. 5.3: *This image illustrates the colour coding scheme for different academic years. It can be seen that the highest student repeated the second year of their studies, resulting in two consecutive yellow cubes.*

5.2.3 Orientation

The year cubes that form the student cuboid, extend the student cuboid in the negative direction of the z-axis. This creates a timeline for the student in the university, extending from their first semester of university to the most recent. Using this timeline that has been created through the direction of extension of the years along the z-axis, we use this as a basis for representing time-dependent and time-independent information about the student.

The time-dependent, consists of the academic year date, the student's yearly overall grade, module grades, and semesters in which the modules were taken. This information is represented on the X and Y axes facing sides of the cubes, to make the information representation consistent with the timeline that has been established before, and maintain a continuity of design in the visualisation as highlighted in the work of *Tory and Möller* [1].

```
var materials = [
    //x facing
    new THREE.MeshLambertMaterial({
        map: new THREE.TextureLoader().load( 'images/term1.png' ),
        transparent: true,
        opacity: transparency_level,
        color: cubeColor}),
    new THREE.MeshLambertMaterial({
        map: new THREE.TextureLoader().load(date_year),
        transparent: true,
        opacity: transparency_level,
        color: cubeColor}),
    //y facing
    new THREE.MeshLambertMaterial({
        map: new THREE.TextureLoader().load(year_result),
        transparent: true,
        opacity: transparency_level,
        color: cubeColor}),
    basicMaterial,
    //z facing
    frontface,
    new THREE.MeshLambertMaterial({
        map: new THREE.TextureLoader().load('images/personal_details_template.png'),
        transparent: true,
        opacity: transparency_level,
        color: "#ccccff"})
    ];

newCube = new THREE.Mesh( geometry, new THREE.MultiMaterial(materials));
```

Fig. 5.4: *This code excerpt shows how UV mapping was used with Three.js MultiMaterial to encode data on the faces of the individual cubes that combined to create the student cuboid.*

The information displayed on the face of the cube in the positive x-direction displays a breakdown of the year cube into terms or semesters. The modules and results for a student are recorded on this face of the cube, which allows the observer to see the grade as well as the timing

of the module. Modules that span one term are located directly under that term heading, while modules expanding both terms are written across the two headings, as seen in *fig. 5.5* below. The side of the cube facing the negative x-direction was used to display the date in which each academic year occurred, demonstrated above in *fig. 5.4*, while the side of the cube facing the positive y-direction shows the end of year result achieved for that year, demonstrated below in ***fig 5.5***.

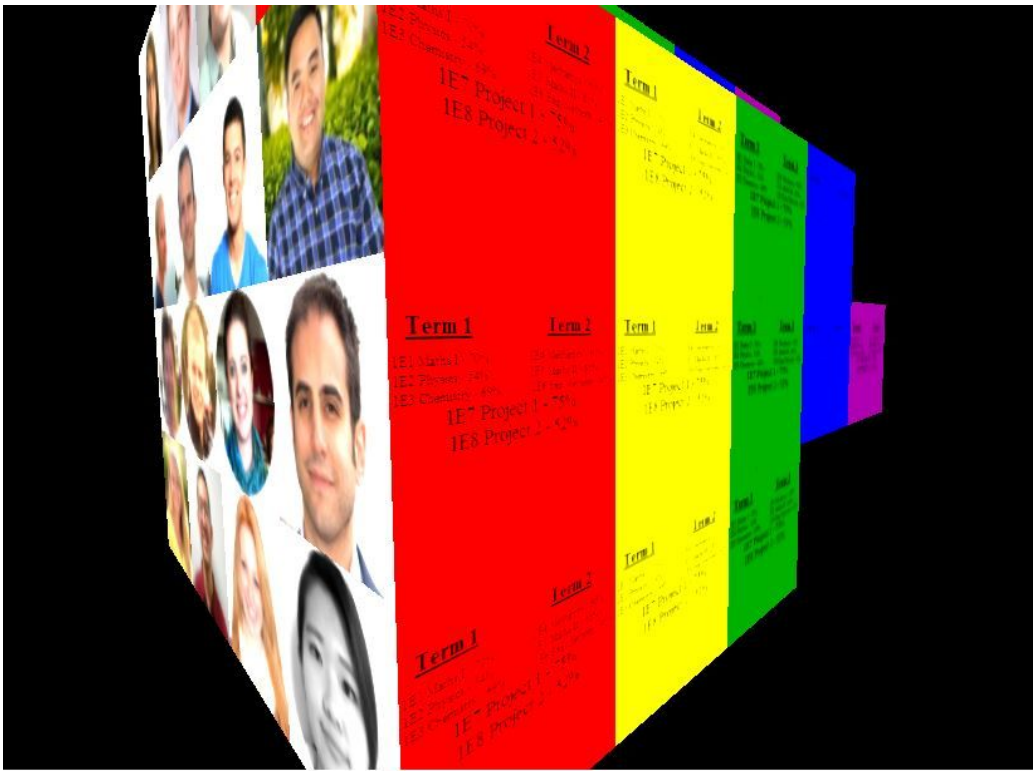


Fig. 5.5: *This image illustrates the semester timeline, which shows student modules and their respective grades.*

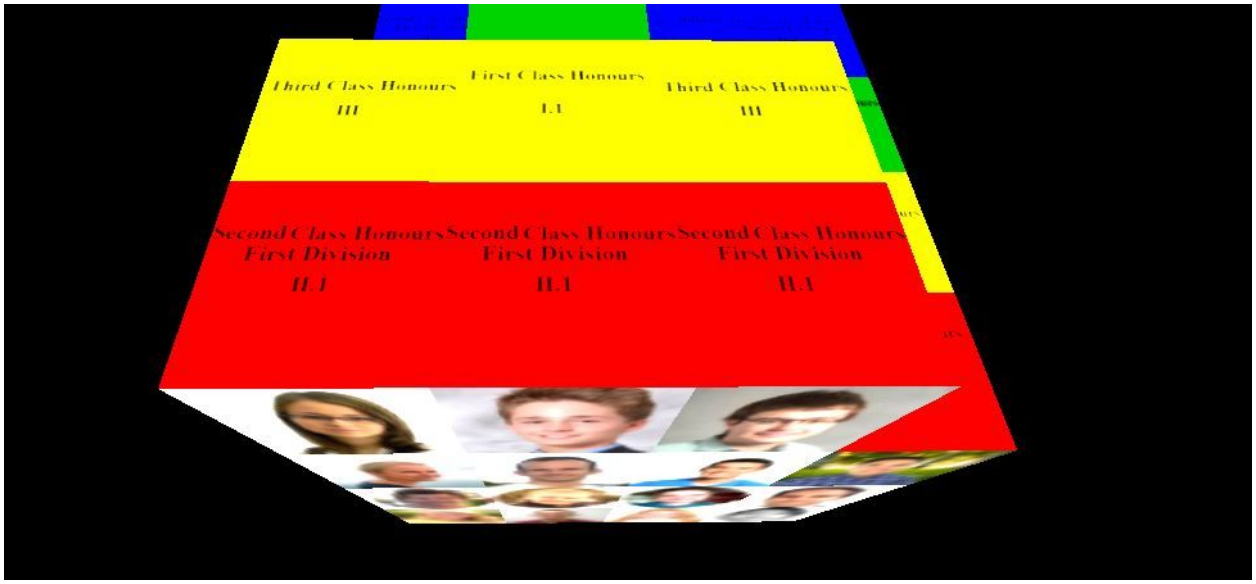


Fig. 5.6: *This image shows the end of year results displayed on the top face of the year cubes.*

The time-independent information is shown on the z-facing sides of the student cuboid. This shows that this information is time-independent as it is represented outside of the time changing information which has been demonstrated to have multiple values for different periods of time, and the time period is reflected within the year blocks along the z-axis. The user is presented with a series of identifying pictures of the students on the cube face in the positive z-direction towards the screen, with personal details being located on the opposite side of the student cuboid, facing the negative z-direction, away from the screen. This helps to reinforce the importance of direction in the system, showing the user that changes in the x and y axes account for moving from one student to the next, while changes along the z axis account for changes in time. The presentation of the student profile pictures side by side also allows for an easy visual search by the user of the system as every student in the class is easily and quickly compared. This could aid a lecturer looking to remember the names of particular students in a class, that they may recognise, but may be still unsure of their full name or student number if they were to try and search for them in a normal textual table presentation of data.

Fig. 5.7: *This image shows the face comparison allowing for a quick visual search to identify individual students.*

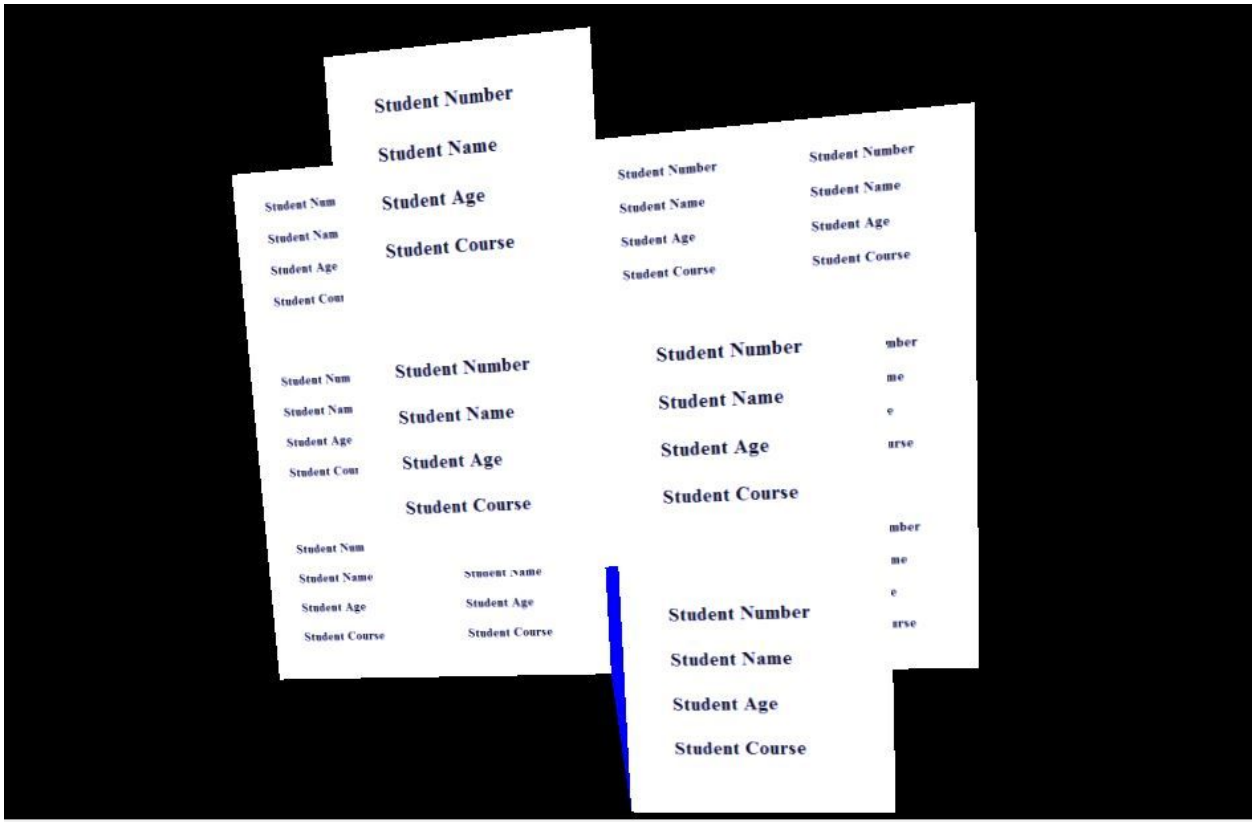


Fig. 5.8: *This image shows the personal details of the students that are displayed on the rear of the student cuboids.*

5.3 User Interaction

User interaction was required as a key part of the system to allow for the exploratory data analysis to occur, as is outlined in a number of the works covered in the literature review. This requirement for interaction to allow for better data analysis was implemented in a number of ways described in the following sections.

5.3.1 Rotation and Zoom

The primary method of interaction with the scene is navigation through the scene using the mouse. This was implemented using Three.js' Trackball controls API. The camera focus in the

scene was always set to be the centre of the student block grouping for the purposes of demonstrating the possibilities of this visualisation system, however in future focus would be allowed to shift to other groupings or individual students. The use of the Trackball controls allowed the users to manipulate the scene to see the class grouping from all angles and distances possible.

```
var cameraControls;
cameraControls = new THREE.TrackballControls(camera, renderer.domElement);
cameraControls.target.set(40,20,-30);

function render() {
    requestAnimationFrame(render);

    delta = clock.getDelta();
    cameraControls.update(delta);
    renderer.render(scene, camera);
};
```

Fig 5.9: *This image shows the simplicity of implementing mouse controls with Three.js due to the TrackballControls library provided.*

The user can rotate the scene in the visualisation by clicking on a point and dragging with the mouse. The scene will rotate in the expected direction and also will maintain momentum through the rotations after the mouse button has been released. The user is also able to zoom in and out towards or away from the centre focal point of the scene using the scroll wheel of the mouse or through a pinch and grab with two fingers on a mobile device.

The zoom function is very important for the visualisation of the student information, as it allows the users to move between a highly detailed area of focus, where there is a lot of detail on a small number of students, in contrast to when they zoom out and see a large amount of information but with little detail, gaining insight on the group as a whole rather than individuals.

These actions are very intuitive as they maintain a continuity from what end users expect and have experienced using other applications, thus reducing the cognitive load on end users and allowing them to focus on their data analysis task. This would be expected to lead to reduced time required for the analysis and more accurate analysis as well.

5.3.2 Variable Opacity Levels

Fig 5.10: *Before the application of transparency to background students.*

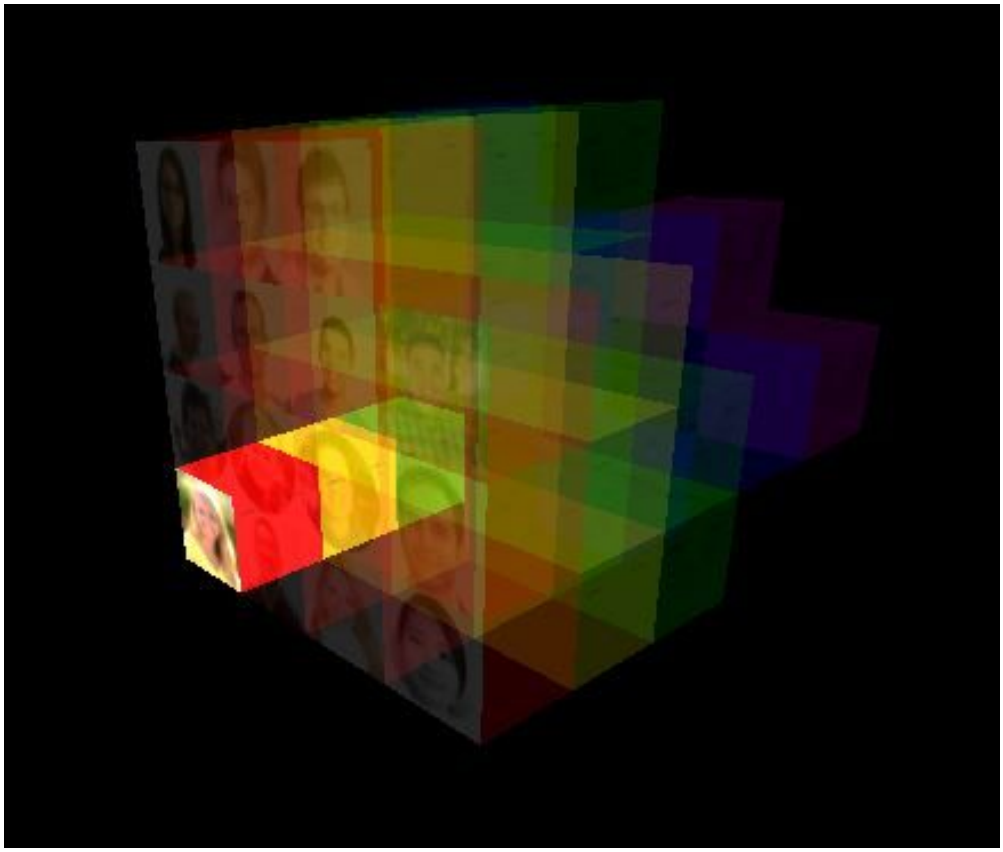


Fig 5.11: *After the application of transparency to background students.*

As illustrated in the images above, the user is able to vary the levels of transparency and opacity of an object. When a student of interest is highlighted by the end user through a method of interaction, they are able to change the opacity of the surrounding students in the group so that they become almost completely transparent. This allows the user to see all of the information on that individual student, as when the students are in a class group occlusion occurs with data that is facing towards the interior of the grouping. However the opacity level change to almost completely transparent allows for the user to overcome this and still see the information that is hidden if they so wish. The problem of occlusion in three-dimensional visualisation systems was highlighted in the literature review in the work by *Tory and Möller* [1]. The change in opacity of the background student objects from fully opaque to semi-transparent was important to allow for a focus-and-context style view as discussed by the authors of *"Designing the Visualization of Information"* [2]. The semi-transparency allows for the user to be able to see all of the

information about the student of interest, while also still seeing the the geometric and colour information of the specific student in reference to the entire group, thus providing the information in context. Providing the information in context reduces the cognitive load on the user of the visualisation as they do not need to remember the properties of the group to see how their selected student compares, instead the information is all in front of them through the use of semi-transparency.

Currently the visualisation system only supports a binary setting of opacity in the background group of students of the selected student. However a future development in this area would be to allow for the user to dynamically adjust the background students' opacities, so as they can see the group with greater clarity, or focus on the individual more. This increased flexibility in the opacity level variation would allow the users to modify the opacity to the level that they require for the current analysis task.

```
//changes the student transparency to the boolean value passed for an individual student in the class
function setStudentTransparency (index, transparency) {
    for (i=0; i < class_group.children[index].children.length; i++) {
        for (k=0; k < 6; k++) {
            class_group.children[index].children[i].material.materials[k].transparent = transparency;
        }
    }
}

//sets the class transparency with the passed boolean value
//except the student at specified index
function demoAllSetTransparency (index, transparency) {
    for (curr=0; curr < class_group.children.length; curr++) {
        if (curr != index) {
            setStudentTransparency(curr, transparency);
        }
    }
}
```

Fig 5.12: *This image demonstrates the functions used to set the opacity of the background students of the selected student by the user.*

5.4 Problems

The main problems that occurred during the course of the implementation were to do with an unfamiliarity with graphics programming techniques, the Three.js library and debugging issues. While implementing features using the Three.js library it was difficult at times to ascertain what was feasible and which methods were depreciated due to a lack of documentation. The documentation also did not always provide sufficient examples on how to use particular methods. The result of this meant that there was an initially steep learning curve, however the library still greatly simplified the overall development process, such as the cube primitives available used as year sections in the individual student profiles.

As a result of the unfamiliarity with many graphics techniques, such as ray casting used for selecting students, and the insufficient documentation for Three.js at times, this meant that a lot of the development work for the visualisation was achieved through exploratory programming. The process would involve developing ideas on paper of how data could be represented, and then attempting to bring to life the design in the visualisation. As a result of this exploratory programming it meant that a large amount of debugging was required for when incorrect or non-existent JavaScript methods and techniques were used. Also due to an unfamiliarity with JavaScript and Three.js, this led to difficult with debugging the program to find error sources.

6. Evaluation and Results

6.1 Final System

The final system shows a representation of a student in a cuboid, combined with other students to form a class cube structure. Students' transcripts, photos and personal information are represented in the system through a combination of visual properties and text. The colour encoding represents the academic years a student has undertaken, while the length of the cuboid represents the amount of time overall a student has been present in the university. The students identifying photo and personal information are encoded on the z-facing sides of the cuboid, outside of time sensitive information represented by changes along the z-axis. The student's academic record for each year is recorded on the positive y-facing side of the year cube, while the modules and their respective grades are encoded on the positive x-facing side of the year cube. The analyst can interact with the visualisation by using the mouse to rotate the class cube to view from any angle, and use the mouse wheel to zoom towards or away from the class, to see more or less detail on the students in the class. Individual students can also be selected by the user and remain opaque while the rest of the class becomes transparent, to allow the user to analyse all the information on a student without occlusion, while also seeing the selected student's cuboid in context of the other cuboids because they have been made transparent and not invisible, allowing for a focus-and-context inspired style display of the student's information.

6.2 Advantages of System

One of the advantages of the developed visualisation system during this research, was in the relative simplicity of the system. This simplicity, of both the controls and the data transformation, allowed the system to be very intuitive and reduce the cognitive load placed on a potential analyst of the student data. The intuitive control system also maintained a continuity

between other similar graphical displays, using a click-and-drag navigation, which is what users are accustomed to using. This lowers the learning curve of the system and increases the ability of the analyst to maintain focus on the current task.

Another advantage of the designed visualisation system was the ease with which individuals students could be identified. The user could easily identify a student that they recognise through a visual search of the class group, as the pictures of the faces of the students are all aligned closely to facilitate this process. Other outliers due to the amount of time they spent in the university could also be easily spotted due to the length of their cuboid relative to other students in the class or grouping. So while the system did not provide a summary of details, it does manage to allow for the easy identification of individual students who may be of interest to the user of the system. The data transformation however could be improved to allow for better identification of students in the centre of the grouping, as the problem of occlusion can hinder the hinder its effectiveness.

6.3 Disadvantages of System

One of the disadvantages of the system is the occlusion of information. This is a common problem with a three dimensional visualisation system as discussed by Tory and Möller [1]. When the student blocks are grouped together into the larger class, it becomes impossible to read the information that is encoded on the sides of the blocks that face towards the centre of the group. Occlusion occurs as in normal three dimensional space, where one object cannot be seen because it is hidden behind the other. Solutions to this problem would be The implemented method for reducing this problem of occlusion was to allow for the user to select a student whose information blocks would remain opaque, while the other students' blocks would become transparent, thus allowing the selected student's information to become completely visible to the user. However this is not a complete solution to the problem, which could be a good focus area for future work on the project, which could include research on using multiple views or data transformations simultaneously

Another problem with the developed solution is the requirement for extra information to understand the layout. Once the layout has been understood, it is easy and quick to perform visual searches to spot differences between the length of student blocks and colours to understand the meanings. This allows for a quick analysis by an experienced user, however inexperienced users may take a short amount of time to understand the layout of the system, what is available, and possible manipulations of the data. As per the recommendations of Forsell and Johansson **[**6**]** extra information should be provided to allow the user understand anything that is not intuitive, to reduce cognitive load on the user of the visualisation system. This is another area that could be improved upon in the future, through the implementation of user prompts and legends beside the visualisation to increase the ease of understanding.

Another shortcoming of the system is that there is no aggregation of information. The only aggregate formed is the visual aggregate of the large class cube made by the combination of student groups. There is no gathering of information based on the students to provide an easy insight into the class, such as the average age of students, gender ratio, or course taking the class ratio. These insights could be useful for an analyst to know, and if the system could provide the user with that information it would reduce the processing required of them. This could be a potential future area of work for the system to expand to, as currently the system is better for exploratory analysis to find individual students, rather than understand the group as a whole, or as a section of smaller groups.

7. Conclusion and Future Work

7.1 Impact of Work

This designed and partially implemented system could prove to be valuable in a university setting. It would be of particular use to any people in the university that needed to take a detailed look at a particular course or class of students, such as a lecturer that was trying to learn more about the students in their current or future class. The visualisation could aid the lecturer to easily recognise and associate students' faces from the pictures, with the data on the student presented in the information, due to an association stored in their visual memory. The lecturer could also easily spot students that had repeated a year who may struggle with the course and thus aid him to change his lecturing approach. The system could expand to reflect students progress with continuous assessment in a class, allowing a lecturer to easily know which students he needs to focus on and perhaps offer extra advice, rather than having to rely on asking a class of students who may be too intimidated to admit they are struggling with their coursework in front of their peers. However this system is not expected to be a replacement for more traditional student database management tools, but to complement traditional tools by providing end users with an intuitive and effective manner of exploring their students' information to gain an insight.

7.2 Future Work

7.2.1 Immediate

There are a number of ideas that would be implemented in the immediate future of this project. The first would be the implementation of ray casting as a method of interaction for the user with the visualisation. This would be implemented to allow the user to select specific students using

mouse clicks when hovering over the student cuboid on the screen. This would work by sending a ray from the two-dimensional screen into the three-dimensional world, and would calculate which object the ray intersects, to know which student cuboid the user has hovered the mouse over. This would be done immediately as Three.js provides example code and functionality for implementing ray casting in a graphics application, making it a relatively simple piece of work to implement.

Immediate future work would also work on upgrading the transparency setting abilities of the system as well. Currently the system only displays the student cuboids at fully opacity, or at an opacity level of fifteen percent. The future work would like to update this ability so that the transparency of the unselected students information could be adjusted on a continuous spectrum from completely transparent to fully opaque. This would be useful as it would allow better individualisation of the visualisation to the user's specific needs. There may be cases where the user finds the partially transparent students to be distracting from the selected student, or the opposite situation could arise where the user would like to see the other students' data more clearly, to see their selected students information in context. The control for this could be implemented through the use of a keypress to increase or decrease the transparency after a student has been selected, however a more intuitive but slightly more complicated solution may be to use an on-screen graphical slider button that would increase or decrease the transparency.

7.2.2 Medium Term

Work in the medium term could look at the improved use of colour in the visualisation. The progression of the academic years could be more easily perceived and more intuitive if the colour changes were upon a perceptually linear scale as discussed by *Tory and Möller* [1]. Using a perceptually linear scale of colours could make it clearer to users of the visualisation that there is a progression from one year to the next, that coincides with the progression of colours that they are seeing from one end of the student cuboid to the other. The colour encoding could also be used for different meanings on different faces of the year cubes, in order to encode more of

the information and make better use of preattentive processing that can be done by users in a visual search. The year grades that are currently encoded as text on the positive y-facing side of the cubes, could take advantage of colour to reduce the cognitive load on the users by using a colour association for better and worse grades that a student has achieved. This could take advantage of pre-existing colour associations as well to make the grade-colour association more understandable, such as using more green colours for better grades, and more red colours for worse grades. This encoding could allow a lecturer to quickly see which students in their class have been struggling when performing a quick visual search.

Another future development within this time frame would be to look at the incorporation of multiple classes or groupings of students in the visualisation. The system currently only supports one class of students, with a variable number of students. However in the future the system would need to allow users to change the class they are observing, so this would be a necessary development task. It would also be of interest to investigate, to try and discover a practical and easy way for users to navigate between classes. This future work could look at displaying multiple groupings of students concurrently, and allowing the users to move the camera between them depending on which class they want to look at, or perhaps have a free flying camera that is not fixed to a particular grouping of students. The work could then also continue to allow users to explore interactive grouping methods, allowing users to select subsets of classes to observe, and also explore algorithms on how to sort or group students based on year, grade or other variables. This would be a useful area of future research as it would make the system more practical for real-world implementation, and also allow for greater customisation of the visualisation to the user's needs.

7.2.3 Long Term

Long term future work for this visualisation system could try to incorporate more end user involved design to benefit from human factors aspects as discussed in *Human Factors in Visualisation Research* [1]. This approach could improve the system by testing the design

hypotheses maintained about the effectiveness of particular visualisation designs and interaction methods, showing what is working in the system and what needs to be improved upon or should be removed from the system. The development process could even extend user interaction to involvement of users in rapid prototyping development. This has been shown to reduce implementation time due to less time being spent on furthering ineffective designs, and also allows for the testing of a wider range of visualisation designs [1]. Thus incorporating user testing could allow for a better system evaluation, and improve future development time, which is why it should be considered an important area of work in the future for this research.

Further long term work could look at the addition of a database to the visualisation system as well. This could look at trying to complete more of the original designed architecture for the system by designing and implementing the relational scheme and tables. This work would involve the use of SQL code to create and populate the database with the records, as well as development on the backend application for retrieving records from the database. The major part of this work would be looking at transforming user requests in the browser-based application, which the research focused on, into SQL queries that would return the necessary information to the application,so that it could then perform the data transformations and visualise it for the users. This work would possibly look at developing a form of table algebra to relate tables to queries as discussed in the review of *Polaris* [3]. This work would be differ greatly from the research that was conducted, as it would focus on the backend of the visualisation system as opposed to the frontend which was the focus of this research. However if completed the system would be much closer to being a complete application, only requiring the completion of the backend server code to host the web service and respond to HTTP requests.

8. References

1. M. Tory and T. Moller, “Human factors in visualization research,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 1, pp. 72–84, 2004.
2. L. Engelbrecht, A. Botha, and R. Alberts, “Designing the Visualization of Information,” *International Journal of Image and Graphics*, vol. 15, no. 02, p. 1540005, 2015.
3. C. Stolte, D. Tang, and P. Hanrahan, “Polaris: a system for query, analysis, and visualization of multidimensional relational databases,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 52–65, 2002.
4. D. Bihanic and T. Polacsek, “Models for Visualisation of Complex Information Systems,” *2012 16th International Conference on Information Visualisation*, 2012.
5. M. J. Eppler and R. A. Burkhard, “Visual representations in knowledge management: framework and cases,” *Journal of Knowledge Management*, vol. 11, no. 4, pp. 112–122, 2007.
6. C. Forsell and J. Johansson, “An heuristic set for evaluation in information visualization,” *Proceedings of the International Conference on Advanced Visual Interfaces - AVI '10*, 2010.
7. N. Elmqvist and J.-D. Fekete, “Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 3, pp. 439–454, 2010.
8. T. Keller and S.-O. Tergan, “Visualizing Knowledge and Information: An Introduction,” *Lecture Notes in Computer Science Knowledge and Information Visualization*, pp. 1–23, 2005.
9. K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, Jan. 2007.
10. C.G. Healey, “Choosing Effective Colours for Data Visualization,” Proc. IEEE Visualization, pp. 263-270, 493, 1996.
11. S. Daly, “The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity,” Digital Images and Human Vision, A.B. Watson, ed., pp. 179-206, Cambridge, Mass.: MIT Press, 1993.

12. J. Lubin, “A Visual Discrimination Model for Imaging System Design and Evaluation,” Vision Models for Target Detection and Recognition, E. Peli, ed., pp. 245-283, World Scientific, 1995.
13. G. Kindlmann and J.W. Durkin, “Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering,” Proc. IEEE Symp. Volume Visualization, pp. 79-86, 1998.
14. J. Bertin, Semiology of Graphics. Madison, Wis.: Univ. of Wisconsin Press, 1983. Translated by W.J. Berg.
15. L. Wilkinson, The Grammar of Graphics. New York: Springer, 1999.
16. M. Ignac, “Pragmatic Physically Based Rendering : HDR.” [Online]. Available: http://marcinignac.com/blog/pragmatic-pbr-hdr/. [Accessed: 18-May-2017].
17. “WebGL - WebGL - OpenGL ES for the Web,” The Khronos Group. [Online]. Available: https://www.khronos.org/webgl/. [Accessed: 17-May-2017].
18. “three.jsr85,” three.js - Javascript 3D library. [Online]. Available: https://threejs.org/. [Accessed: 17-May-2017].
19. M., “mrdoob/three.js,” GitHub, 17-May-2017. [Online]. Available: https://github.com/mrdoob/three.js/. [Accessed: 17-May-2017].
20. J. Etienne, “Basic Example of a Cube with Three.js”, GitHubGist. [Online]. Available: https://gist.github.com/jeromeetienne/1106726 [Accessed: 17-May-2017]
21. F. Soper Mac Cafraidh, (2016, August 8). Internship Technical Report - Deloitte [Online]. Available e-mail: sopermaf@tcd.ie Message: Get Internship Technical Report.