AC40001 Individual Project Mid-term Report

Multi-Dimensional Web Plotter

Sameer Al Harbi - Student | Dr. Iain Martin - Advisor

January 27, 2023

1 Introduction

Data visualisation is an important step in the data analysis process- with it's importance only rising with the size and complexity of the data involved. This project aim's to take advantage of the performance offered by graphic processors and the accessibility of the web to create a Web Application that allows a user to explore large, multidimensional data set's without lengthy downloads, setup or specialist knowledge. While still having the tools needed to identify interesting trends in the data that would aid further analysis and generally help with understanding the structure of the data.

2 Background Research

Before any plans for the project were begun, it was considered that a study of the current available solutions would be prudent to identify opportunities to create a unique Application that addressed the pain points of the market at the current time. It was found that there was a large number of different solutions each with their own advantages and disadvantages for different scenarios. In the appendix A, there is a a summary of some of these solutions that each highlight some important aspect of the overall market and have directly motivated the design of the Application being developed by the Student.

In general, it was found that most solutions are highly technical and need at least some degree of programming knowledge to use. A rough relationship would be that the complexity of using the solution and it's capability is inversely proportional. Feature rich solutions need programming experience while the one's that don't, have more limited features, and are more likely to be not free to use. This relationship was found to be important because there are non-data or software focused markets [4], and thus users, where visualisation solutions are needed but programming experience is not as widely common.

Most of the best solutions were also analysis solutions first and foremost with visualisation as one of the features. Of the tools that did focus on visualisation, they also suffered from the same relationship as above.

Taking all this into consideration, a number of key points were identified on how to structure a new development project to create a valuable solution from the market's point of view:

- Focus on accessibility first and foremost, including an easy to use interface that does not need programming experience. This also by extension means the Web is an ideal platform to simplify installation requirements, and to tap into the rich UI feature set of browsers.
- Focus on the visualisation aspect, but either ensure that adequate analysis tools are available or importing and exporting data is easy and straight forward. Visualisation is often only one step of a multi-step analysis project.

On the other hand, some thing's to avoid are:

- Focusing on the feature's but not accessibility. It is unreasonable to try to beat the current market players on features alone- You can't get more feature-rich than a programming language such as Python, which is arguably one of the most used solutions on the market, See Appendix A.3 for more information. It is important instead to create an alternative that is powerful enough, but makes the process of data visualisation much easier.

All designs and plans for this project follow these points. Beyond market research, the features of these solutions themselves have been considered and used to inspire the development process, Scatter plots were found to be a common chart type that were conceptually easy to render even at beyond three dimensions- making them the perfect candidate to focus the first version of the application on, See 3.

3 The Application

The current client brief for the first version is described below. See 4 for more info.

"*A Client side only Web-Application is needed that allows a user to import a .CSV file that is plotted on a 3D Scatter graph with labelled and accurate axis values. The User should then be able to navigate through the graph by zooming into the data, moving the application view around the scatter graph in 3D, and rotating the scatter graph*"

This brief has already been analysed, and user stories created that all together make up one feature set. See the MVP Feature Set Appendix for further info C.1.

4 Current Progress

Beyond research, The other key tasks undertaken in semester 1 have been designing an iterative first version, deciding on tools and methodologies, prototyping and the start of development. More information on each of these steps is covered below:

4.1 Tools and Methodologies

Choosing what kind of application to make and the tools to create that application was an important consideration for this project. It was decided that a web based client Application should be made in line with the points identified in the research phase. With the web set as the target platform, a suite of main tools was shortlisted for development. Those are:

- **TypeScript** as the programming language, adds a type system and other useful language feature for graphics development which are otherwise missing in JavaScript.
- **WebGL**, as the graphics API, Alternatives would have been either the newer WebGL2 or even a higher level framework such as Three.js. But, It was decided to forgo frameworks for greater flexibility and the experience of working with a low level API. WebGL2 was not chosen due to limited compatibility in devices.
- **Parcel** as the compiler and build tool, Parcel was chosen due to it's ability to greatly improve the developer experience (and app performance) through compiling and bundling of assets into a single .JS file + a handful of helper files that could easily be hosted statically. This has allowed the the application to be separated into multiple .JS files and have shader and 3D model files be separate from the code during development.
- **Node.js + npm** as the development server and package manager.

Services were also chosen to help with development, those are:

- **GitHub** as the central code repository, chosen due to personal familiarity
- **Digital Ocean, App Platform** as the hosting provider for releases, chosen over GitHub pages due to more flexibility in case other resources will be required in the future, and personal familiarity.

In terms of methodologies, it was decided to structure the project in an agile way. Specifically PXP as as described in this paper introducing a one-person variant of the traditional XP methodology[3]. This methodology was further adapted to this project with major changes being a more manual approach to developer testing (due to limited opportunities in automatically testing graphics programming), having multiple briefs for each version of the application, and a slight restructure of treating user stories as tasks themselves. User Testing is also added as a post-version task.

4.2 Design

Designs are done as described in the PXP Paper[3]. A client brief is created by the student and this brief is then analysed to create a set of user stories which are subdivided into feature sets. Iteration's are then planned to complete each user story. Only the brief for the first version consisting of one feature set has been created so far, which can be seen in the Application section above. Designing the brief for each version is done to allow user feedback to be considered.

4.3 Prototyping

After the tools and methodologies were chosen, the prototyping phase was started to identify any shortcoming's or difficulties for the project at hand. The first prototype was to setup the development environment and get the WebGL rendering to work. It was successful and minimal difficulty was encountered.

The next prototype was to get a full 3D renderer made, with the ability to load in and display 3D models. This prototype naturally took a

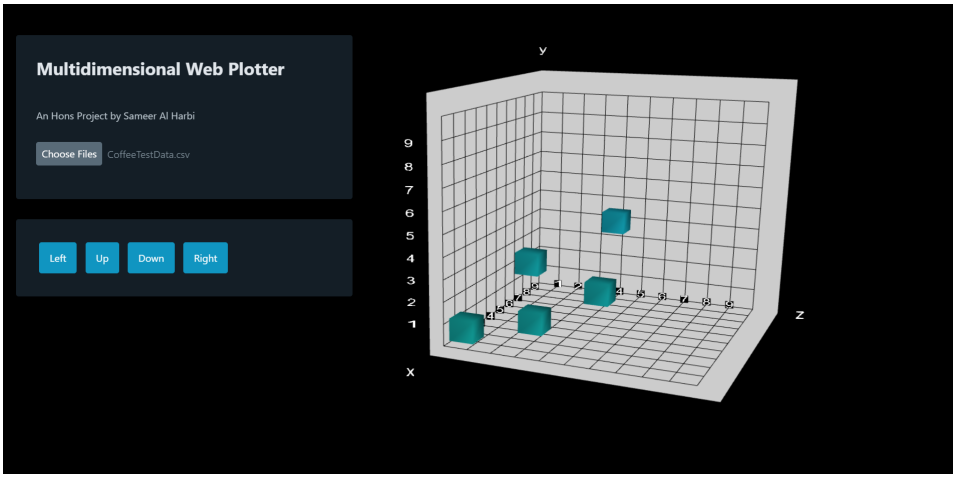

Figure 1: The Current state of the Application at the end of Iteration 2

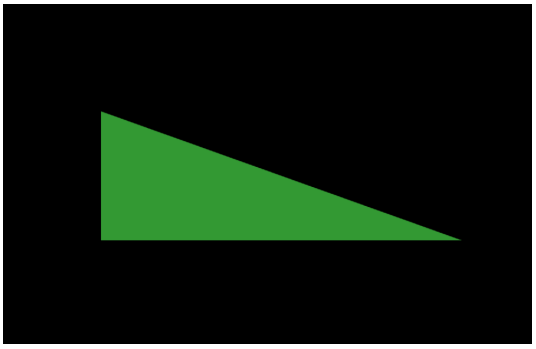

Figure 2: First Prototype showing a working WebGL Context and a simple 2D geometry

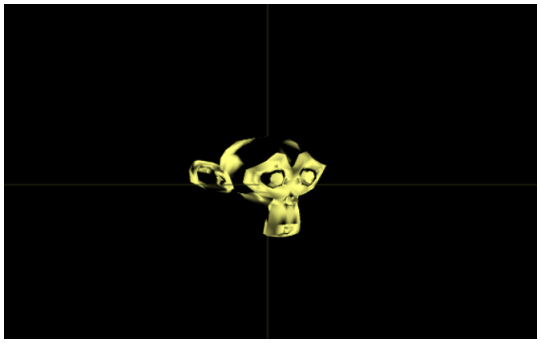

Figure 3: Second Prototype showing a loaded .obj model

bit longer- mainly due to questions on how best to structure the code. But it was also successful and was refactored to start the actual application development from.

4.4 Start of Development

By the end of the 1st semester, the project was ready to be started- and at the time of writing 2 iterations were completed. An iteration retrospective for the first iteration is available here in the appendix B- with the second iteration still being in the process of finalisation. As for the Application, completed user stories can also be viewed here in the appendix C.

5 Future Development

For future development goals, it is planned to develop the application to version three, meaning three briefs in total highlighting large expansions. Upon completion of the first two versions it is also planned to have a user testing phase too. For more details on this see the Gantt Chart Attached here D. Most major potential roadblocks have already been mitigated through prototyping in semester 1, those were mainly on difficulties encountered when using new technologies and design questions. But some potential roadblocks can occur if some unexpected user stories take longer than expected or require considerate refactoring. To mitigate this risk, the project is planned to be completed 14 days earlier than the deadline. If particularly large risks occur and more extreme measures are needed, the 3rd version can be cancelled or have it's development time shortened by creating a brief with less features needed. Report writing will be done concurrently with development, with a bit of time at the end of the project to finish up any post development write-up. Which should be minimal since the bulk of the report would have been written by then.

Conclusions and Personal Statement by the Student

The Project is going quite well in my opinion. I should have the most vital features done quite soon, and be able to start planning for user testing and future additions after that. Getting the first feature set is a priority and I'm glad to have made so much progress on it over the winter vacation. Doing prototype's in the first semester turned out to be pretty useful to get used to developing graphics on the web, which I've never done before without a framework, and at this point I'm pretty comfortable with the development process, meaning large roadblocks are going to be less likely.

References

[1] Stack Overflow. (n.d.). Stack Overflow Developer Survey 2021. [online] Available at: https://insights.stackoverflow.com/survey/2021section-most-popular-technologies-programming-scripting-and-markup-languages.

[2] Stack Overflow. (n.d.). Stack Overflow Developer Survey 2021. [online] Available at: https://insights.stackoverflow.com/survey/2021section-most-popular-technologies-programming-scripting-and-markup-languages.

[3] Agarwal, R. and Umphress, D. (2008). Extreme programming for a single person team. Proceedings of the 46th Annual Southeast Regional Conference on XX - ACM-SE 46. doi:10.1145/1593105.1593127.

[4] Which industries use data visualizations as part of their service – and who is next? (Matthias Tratz). [online] Available at: https://www.linkedin.com/pulse/which-industries-use-data-visualizations-part-service-matthias-tratz/ [Accessed 26 Jan. 2023].

A Appendix: Current Market Research

These are short summaries of each important solution on the market

A.1 Gnu Plot

Gnu Plot is a free to use command line program for generating graphs on multiple platforms and as a plotting engine for other programs. It has been actively developed since 1986 and is the oldest of the tools mentioned here. It is highly customizable and includes a huge number of different plots and styles of visualization. It also can be run on many platforms and export visuals to different formats. Although being a command line tool, it requires specialist knowledge to use and it is expected that the user would read documentation to learn how to use each of the many options available. Having a lot of customizability in graph options similar to gnu plot would be useful to expand the kind of data that could be displayed.

A.2 WolfRam

WolfRam is a language and set of tools used for the computation of data. One feature of the language itself though is it's integrated ability to display 3D plots of data out of array like data points. Further effects can also be applied to create other visuals. This also requires programming knowledge. A scriptable design for visualisations options might be a good way to structure the code where multiple combinations are possible.

A.3 Python

A general purpose language would usually be too vague to identify as an option, but an exception is made here mainly for two reasons. First- Python is arguably one of the leading languages for Data Science and Analysis applications with a wide range of supporting libraries. The Stack Overflow 2021 Developer Survey ranks Python as the 3rd most used language [1]- with multiple data libraries on the most used libraries and frameworks list [2]. As such, it wouldn't be representative of the market at hand to omit it. The second reason is that there are too many python libraries that could be listed here, Some focusing on the visualisation itself while others supporting the function of the former in some way- but all of them can be broken down into the same takeaways as if we consider the whole platform. Those are the following:

- Python is a programming language- As such, there is a high difficulty curve to use it and requires programming knowledge. Although with the large option of libraries it is certainly easier than writing the application you need from scratch.
- The large number of libraries means most tasks can be done through minimal code- this naturally includes visualisation of data too through a library such as matplotlib or one of many others.

- If something is not implemented by a library then it is straightforward (but not necessarily easy) to make it yourself. You cannot say the same for an option that offer's less control to the user.

A.4 Tableau

Tableau is a visual analytics platform that was founded in 2003 with the goal of improving the flow of analysis and making data more accessible. To that end, they have a number of products to help drive business decision making. Some products of particular interest are Tableau Public, which is a platform to share nicely formatted data charts and a no code creation tool. There's also Tableau desktop which gives a drag and drop interface to analyze and plot data from a connected data source. This is the first entry on this list that does not require at least some programming experience- but it is much more limited than the other options, and is not free with a licence required. The easy to use interface of these products provides a great example of what's possible for a data visualisation application.

A.5 MATLAB

MATLAB is a programming and numeric computing platform which contains the ability to create graphs for data visualisation through built in plots. But, MATLAB truly shines in its data analytic ability. Allowing the use of multiple different built in functions and libraries to be ran on data combined with the MATLAB language itself which is a fully featured programming language built for data analysis. Alternatively, a UI based interface is also available to create data visualisation. But MATLAB is not a free tool and requires a licence to use. Great split of option between an easy to use UI, and the alternatively more power programming tools.

B Appendix: Iteration 1 Retrospective

B.1 Summary

During the first iteration of the project development. The code base from App Test 2 was taken as a starting point for creating the application. In total, two user stories were completed rated at a difficulty of 5 and 1 for a total velocity of 6.

B.2 Key Points

- This was unsurprisingly a rough start- owning to the fact that this was the first time that PXP was used and generally as an iteration without any previous other to learn from.
- Automated testing for webgl is really difficult to get right, remove it from the development process for now and focus on writing code.
- The do "Simplest thing that works" didn't turn out too well- lot's of factoring overhead had to be added to fix code that although working, had to refactored else any further additions would take exponential amount of time to implement. Allow refactoring tasks to be raised at any point not only when development of a task or story is complete.

C Appendix: User Stories

C.1 MVP Feature Set

These are the user stories that have been completed for this feature set, the number following each story is the relative expected time cost needed to implement the story. 1 is minimum and 10 is maximum.

1. As a User, I want the application to have easy to use on screen controls for interacting with the application - 3
2. As a User, I want to be able to view a scatterplot of data set against an axis with accurate scales - 5
3. As a User, I want to be able to import a dataset into the application to graph it in a 3 dimension scatterplot - 2
4. As a User, I want all interactable parts of the Application to be visible at all times - 1
5. The Axis should scale with the values shown - 5
6. As a User, I want to be able to rotate the 3D Scatterplot around all 3D axis individually - 4

These are the user stories that are yet to be completed,

1. As a User, I want to be able to navigate around the generated graph in 3D space while having the axis stay accurate - 5

2. As a User, I want to move the 3D view of the scatterplot in 3D using on screen controls - 2
3. As a User, I want to be able to zoom in and out of the 3D Scatterplot - 3
4. As a User, I want to be able to view instructions within the application on how to use the application - 1
5. As a User, I want to be able to access the application on landscape screens of different sizes - 1

D Appendix: Gantt Chart

Note: User testing + design is done concurrently. As feedback comes in it is used in the design of the next version.