The Development of a

Multidimensional plotting program

**Max Fyall**

**ID: 180011724**

**AC40001 Honours Project**

**BSc (Hons) Computing Science**

**University of Dundee, 2021**

**Supervisor: Dr. Iain Martin**

**A screenshot of a computer

Description automatically generated with medium confidence**

***Abstract* -** *The abstract (or ‘executive summary’) is an important part of your report. In essence, it is a summary of the purpose, methods, findings, and conclusion of your project. It should be no more than 200 words. It should be clearly and concisely written. Provide only the most pertinent information, avoid citing references and include a brief statement of your main conclusions.*

# Introduction

Plotting programs (Information Graphics Software) can be used for visualising data in an intuitive manner. Data visualisation is a critical process in the understanding of large complex data sets. It is commonly one of the first stages in the analysis of a big data set. There are various software applications that provide tools for data visualisation, a few examples are, GNU-Plot, Tableau, MATLAB etc. Each application has their own style (i.e. GUI interface, Command Line interface), but they all provide some form of data visualisation. The majority of software predominantly uses graphs and charts 2-dimensionally (2-D) . A simple and effect way to convey given data sets, this is the most common method for graph plotting.

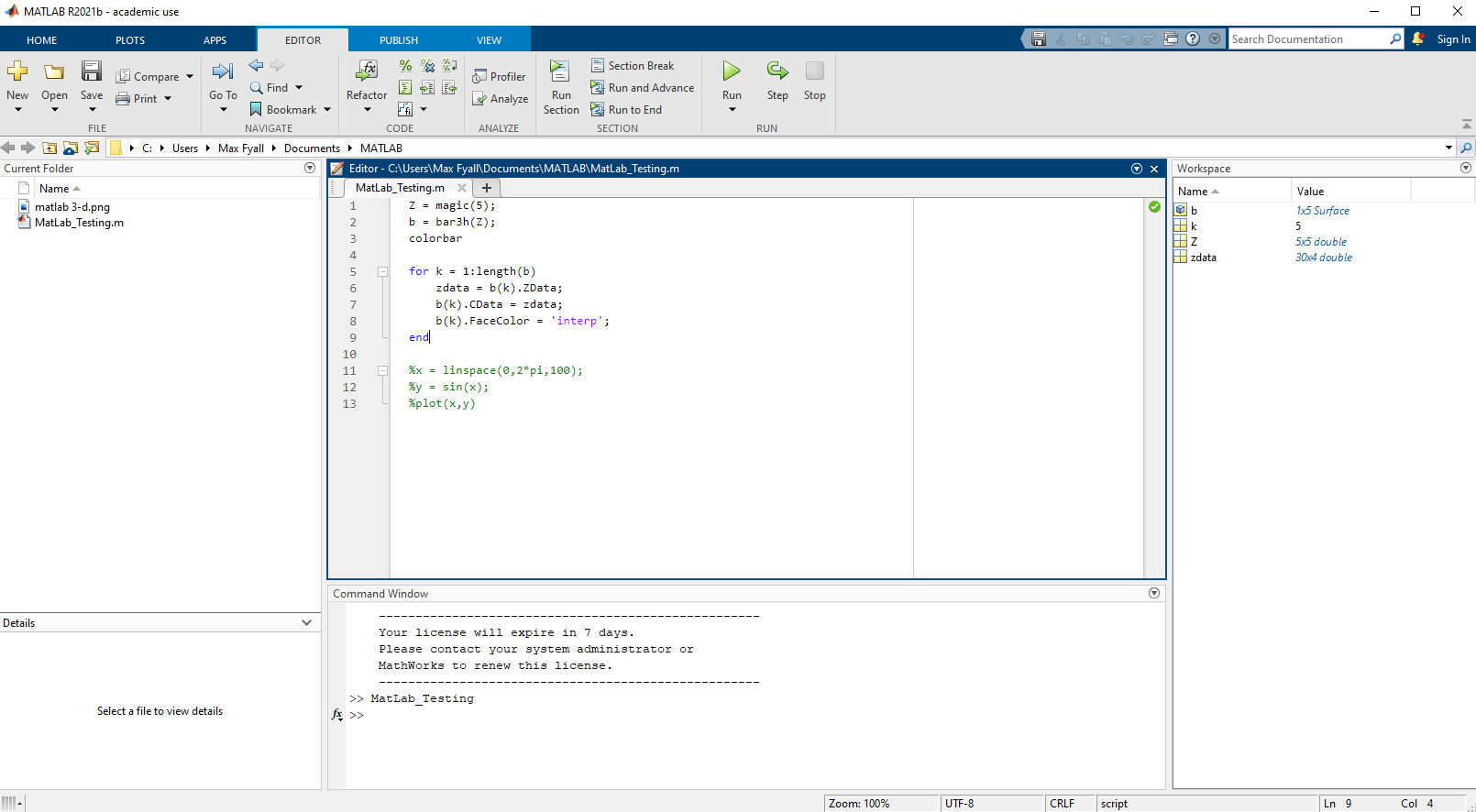
Not so commonly used in better-known software is 3-dimensional (3-D) plotting. Similar to 2-D plotting with the only difference being the addition of another dimension (depth). This dimension can be used to great effect when plotting in 3-D. Different perspectives and a better understanding of data (and subsequently much better analytical results) can be gained through correct use of 3-D plots. However, in some cases, they are often overshadowed in favour of their 2-D counterparts. Why is this the case? The short answer being the advantages a 3-D plot gives, mostly, does not warrant using it over a 2-D plot. They are not helped by the fact that they are more complex than their 2-D counterparts as well. If this is the case, what can we do to improve the 3-D plotting experience? It is with this in mind that the researcher aims to use 3-D graphics technology to develop a prototype multidimensional plotting program that enhances the 3-D plotting experience.

# Background

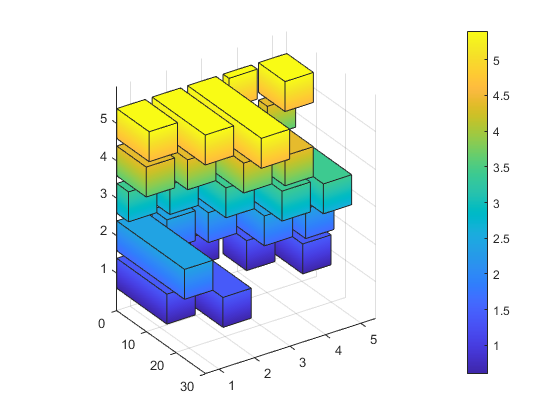
As mentioned previously, there are multiple pieces of software geared towards data visualisation. Each piece of software has their own niche with specific users since each user will have their own set of requirements. For example, Tableau is design towards a Data analyst and Statisticians whereas something like Matplotlib is geared towards Computer programmers since it’s a library for the programming language python. The purpose of this section is to look at existing plotting software, examining how they work to understand the best approach when designing a program of this nature.

Existing Software

Modern graph plotters incorporate both 2-D and 3-D plotting tools. For the purposes of the research, the tools for 3-D plotting will be investigated to understand the features that are present for 3-D plotting. MATLAB in one example that incorporates 3-D plotting very well. MATLAB itself is a programming language used inside the MATLAB software. Using the MATLAB language it is possible to plot functions and data in both 2-D and 3-D.



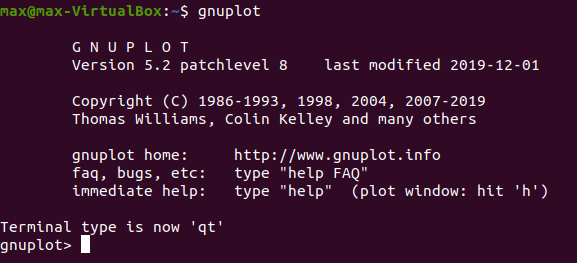
*Figure 1: MATLAB GUI*



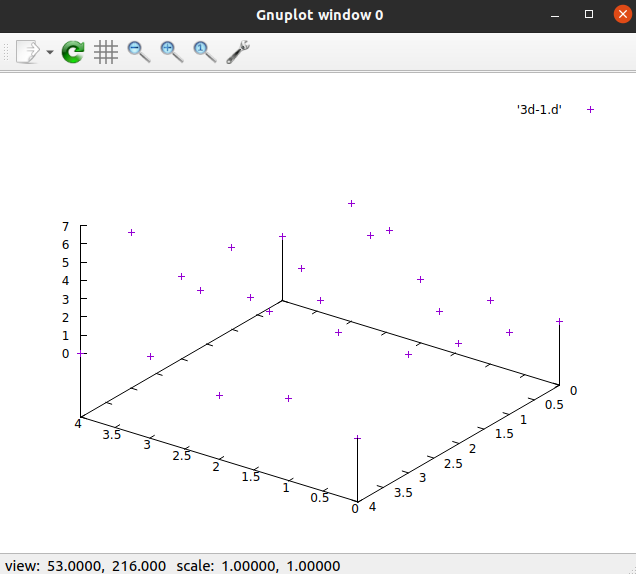
*Figure 2: 3-D Plot using MATLAB*

Upon clicking “run” with valid plotting code, a window will appear with the completed plot as shown in Figure 2. From here the user can use the mouse to navigate around the graph to gain a better perspective of all aspects of the graph. This is very intuitive and is necessary when viewing certain parts of the graph as seen in Figure 2. It is something that has been taken on as inspiration when planning the prototype due to the intuitive nature of it and the benefits it brings to viewing graphs. Whilst this is all great, there is a problem. The software relies on the user having some form of coding knowledge. Having little coding knowledge will likely see most users searching online for how to acquire and accomplish tasks. This is fairly easy to do since there are multiple forums regarding MATLAB. However, this could turn some users away since they could be looking for an easy intuitive system that they don’t have gradually to get the hang of. With this in mind, one goal of the project is to make the design/functionality intuitive for most users to use. This allows for quicker access to the things end users want the most… results.

Another common plotting program mentioned when discussing this particular software is GNU-Plot. Inside the operating system Linux, GNU-Plot incorporates a mixture of command-line and GUI interfaces (See Figure 3 & 4). Meaning you insert commands via a terminal and your resulting graph is displayed within a separate window using “Qt”. It functions very similarly to the MATLAB software in that the software interprets an input and displays a result based on this input. The resulting window also shares the mouse control feature found in the MATLAB software as well as looking somewhat similar (See Figure 4).

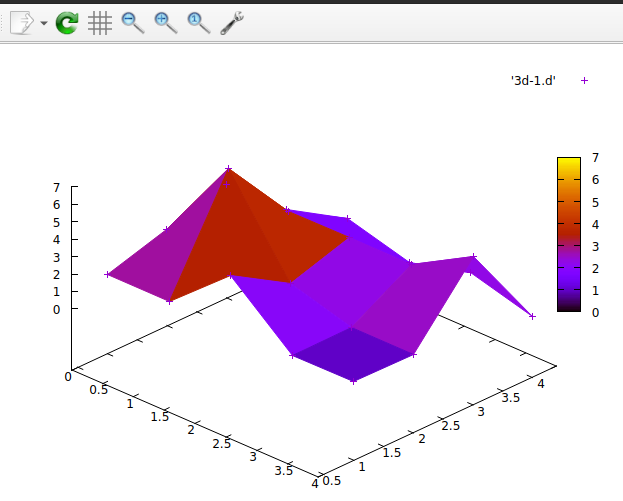


*Figure 3: GNU-Plot Terminal Interface*



*Figure 4: GNU-Plot 3-D Scatter Plot*

GNU-Plot is a great example of command interpreted plotting. It would be a good basis if the project was to be script/command based in a terminal. It is fairly simple to plot from data files as long as you know where your file is store since this could cause file-finding issues. The main problem with GNU-Plot is the overall complexity for plots. A lot of time spent using GNU-Plot is usually looking online at the documentation to find a solution. As we saw with MATLAB, this not ideal for most users. This issue is not helped by the fact that GNU-Plot is mainly used in a Unix environment (i.e. Linux distributions). GNU-Plot comes with most Linux distributions by default, this can be handy for any Unix users since there is no install required. However, this does mean if you are on Windows you need to install a Unix environment (e.g. MinGW, WSL) just to get it running which not a lot of novice users would not be able to achieve. The plots themselves are also not the best looking 3-D plots (See Figure 5). Compared to MATLAB, the plots are considerably less in terms of graphical quality. Using the 3-D graphics libraries and the processing power of the GPU, this will be an area that the project aims to show can be improved.



*Figure 5: GNU-Plot 3-D Scatter Plot with fancy feature on*

# Specification

You might choose to devote Section 3 to a specification of the problem (such as requirements, user stories), and an explanation of how you arrived at this specification. You can describe an initial work schedule, including an overall project plan with time scales and deliverables. You can summarise your development process and methodology, with a justification of the methodology selected.

# Design

You should include descriptions of the main design choices you made at the outset of the project, e.g., in terms of the tools that you adopted, the frameworks, the environments, the programming languages, usability constraints or standards, and so on Did algorithms or data structures have to be selected? Did you have to produce any initial software designs, e.g., e.g., data models, E-R diagrams, UML designs. Design decisions and trade-offs should be described, including consideration of alternatives and a justification for the choices made. Sometimes, the justification may be because of constraints on the project, e.g., the learning curve required for certain technologies and the feasibility within the project timescale. In other cases, there may have been a range of equally comparable tools or technologies that you could have selected from. In which case, why did you choose one over the others?

It is recognised that some design choices may be more, or less applicable to some projects than others. For example, a project which is developing and benchmarking a new image processing algorithm may have less concerns about user interface design, but the latter could be important to other projects. Therefore, please present information which is pertinent to the needs and expectations of your own project.

It is also recognised that design choices may evolve or change during the project, as you discover more during the implementation stages. Therefore, you may need to approach the Design section in a couple of ways. You could present it as a record of what was relevant at the outset, prior to implementation. You can then discuss any changes that occurred during the implementation section later. Alternatively, you can present information about the design choices that you made at the outset, and the subsequent changes to these during the later stages of the project together, at the same time, in the Design section. The choice of which approach to adopt may be down to your own personal preference and/or based on the advice of your project advisor too.

It is also recognised that some projects may have their technology choices pre-ordained from the outset. I.e., there is a specific platform or framework which has been stipulated by the project advisor at the outset and/or which is just the single, natural choice for projects in that area. In such cases, it is still important to outline what these constraints or choices are. It such cases, it may still be feasible for you to explore or consider alternatives.

Finally, it is also recognised that some design choices may subsequently depend on others. For example, if you had decided that C# or ASP.NET was going to be the most appropriate language or environment for you to use, it could be natural for you to conclude that Visual Studio would be the relevant development environment to use too. However, there could still have been other options to consider in these circumstances. Ultimately, you are just looking to provide a robust and comprehensive discussion of your choices, rationale, and justification.

There are occasions where it might be helpful to refer to an equation, such as Equation (1).

(1) *E = mc2*

More generally, you will almost certainly want to use some figures or tables throughout your document, such as Figure 1 and Table 1 below. These should include captions and cross-references within your text where you wish to reference to them. In addition, when you include a figure, screenshot, graph, diagram, table, or similar item into your report, it is important that you explain to the reader what the figure shows and/or what you are attempting to draw their attention to or emphasize. For example, see Figure 1 below which shows a diagram of an extremely important pattern.

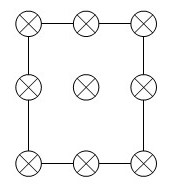


Figure 1. A diagram of an extremely important pattern.

At this point in your report. If you didn’t say any more about Figure 1 above, the reader would have no idea why the pattern it contains is important or not. Please say a little more about what you need the reader to know, or what you wish to emphasize, highlight, or draw their attention to. Please see the example below in relation to a table.

Please see Table 1 below which presents the initial results of performance testing using method A. For each data set, it shows the error rate (%) and time (seconds).

Table 1. Performance of Method A: Initial Results.

|  |  |  |
| --- | --- | --- |
| **Test data** | **Error rate (%)** | **Time (sec.)** |
| *Set 1* | 70 | 3.1 |
| *Set 2* | 74 | 8.0 |

As Table 1 shows, whilst Set 1 and Set 2 have comparable error rates, the performance, in seconds, of Set 2 is much slower and this was deemed to be impactful to the requirements of the machine learning algorithms to be employed.

NOTE: please remember to include Alt Text for any images and Tables that you use in your report.

# Implementation and Testing

You should describe the important aspects of implementation, testing, and debugging that you went through to produce your system. You can structure this in different ways, depending on the development methodology adopted and the needs of your project. You may wish to start with a review and overview of the main features to be implemented and a general, architectural overview of the system. You may then wish to walk through the major features, components, or sub-systems that were created, one after another. These could be sub-sections in your report, e.g., Feature X, Feature Y, etc. Or you may wish to present a time-based review of the implementation process, according to the stages you went through in your project plan. Indeed, if you have adopted an Agile approach, you may wish to structure your discussion around the various Sprints that were undertaken. In your discussion, highlight any important features that were implemented, any major problems that were encountered, and the workarounds that you produced. Your aim is to convince the reader that you are technically competent and that you are capable of problem solving and adapting to needs of the project. The amount / extent of technical contribution is also being assessed and the extent to which you have been able offer original ideas of your own. Regarding the amount of technical contribution. For example, a basic website, with a few, static pages is likely to be rated somewhat poorly. Instead, one would expect dynamic content, a database, more complex code and problems being solved, additional considerations for accessibility, usability, security, etc.

Regarding the implementation section. You may wish to illustrate your discussion with diagrams, or code snippets, that offer additional insights into your work or achievements. You may wish to emphasize user-centred processes, where applicable, and how the system evolved during implementation. For technically oriented projects, it is understood that you may wish to focus more on the performance, accuracy, reliability, or precision in your outcomes, including benchmarking against the work of others. For an additional layer of sophistication, any project can consider additional non-functional aspects of the system which are applicable, e.g., security, scalability, performance, usability, accessibility.

Later in your report, there is a related section: Description of Final Product. This later section is focused around providing a summary overview of your finished product. In contrast, the implementation section focuses on the stages that you went through to achieve and deliver it. There may be some areas of overlap, e.g., when you discuss the implementation of a particular user interface component, and you wish to use a screenshot to highlight the implementation choices made. Meanwhile, it turns out that a similar screenshot is necessary later in the Description of Final Product section, where you are simply presenting what the key aspect of the interface looks like. That is OK. There is just a difference of emphasis here.

For additional sophistication in your implementation, you should consider the use of software testing techniques, e.g., unit testing or similar. If so, the markers would need to see evidence of their use, e.g., in your source code or similar. In addition, you could consider traceability back to your original requirements, and verification or validation that they have been achieved.

As noted above, you may wish to include snippets of code in your report, to accompany your discussion of the implementation. Commonly, these may be included as screenshots of the relevant portions of code. It is best to keep these focused on specific areas of the code, e.g., it may be a specific method or a section of a method. For example, we are developing a web-based system which has a sequence of code for iterating through groups of product items. There is perhaps some reason why this code is noteworthy, e.g., it illustrates a novel approach or solves a tricky problem or is just something you are pleased with. Having discussed the feature, we wish to show a code snippet too. An example of this is below, e.g., please see Figure 2. Code Snippet. Iterating product options. below which illustrates the routine that was implemented to address this challenge. In the code snippet, you will see how the product items are iterated to complete the relevant basket page for the user.

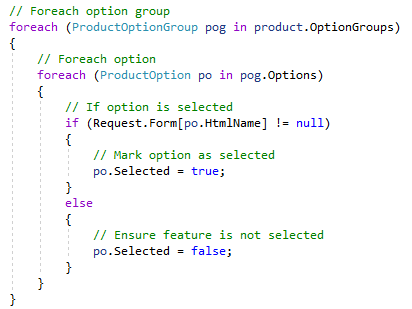


Figure 2. Code Snippet. Iterating product options.

Sometimes, you find that you need to include a larger image in your report that doesn’t fit easily into the two-column layout. In that case, you may be able to use Section Breaks in the document, to temporarily switch to a single-column layout and back again. For example, at this point in the report I have inserted a Continuous Section break immediately below this paragraph. Inside the new section, I have modified the layout to be single-column instead of double-column. I have included an example of a figure that spans the entire page. In this case, the figure is shown on the next page, because it doesn’t fit here. This has left a little but of a gap here on this page. This can be a common problem. Sometimes you can position things in such a way that the gaps are minimised. It is also worth noting that you don’t always need to place a figure immediately beside the text that it refers to. You could place a figure on the next page instead for example, whilst you continue to refer to it here, and afterwards. That way, you can fill up as much of the text here as you can too, without the need to have the figure placed in between. It is also possible to have several figures placed together on one page, and these can be referred to from relevant locations in your text. If you have large images that are impossible to fit into the report without being legible, you can include images in your appendices too.

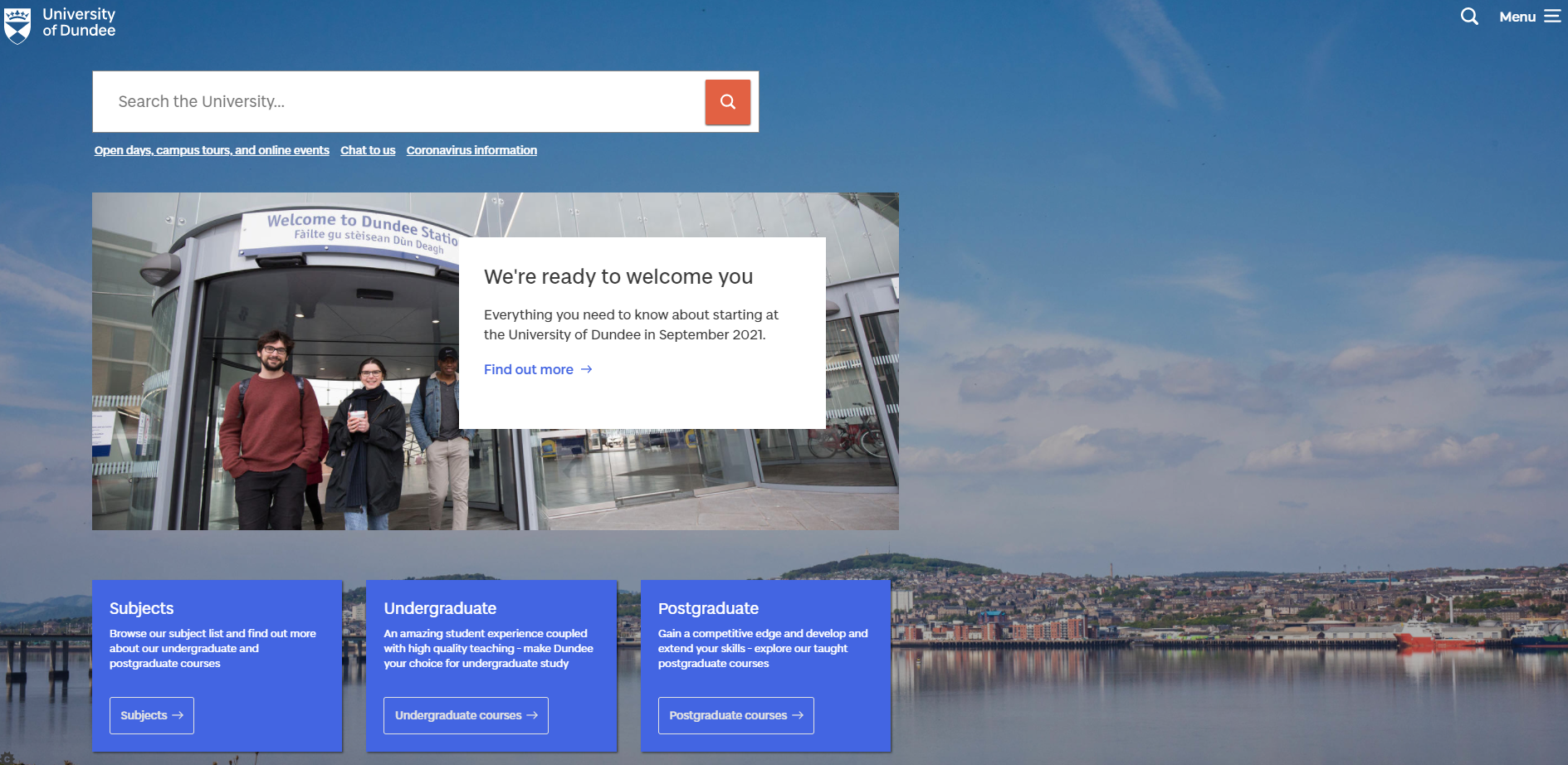


Figure 3. Random Image. In this case, a screenshot of the UoD website.

Now that I have finished what I want to do in this section, I have placed another continuous section break immediately after this sentence, which ends this single-column section and takes me back to the double-column layout afterwards.

# Evaluation / Testing

You must evaluate your system. This will be done in different ways depending on the project. For example, if you are developing a web application or app, it is common to do user testing and you may wish to seek feedback and comments from end users through interviews or questionnaires. If your project has a technical, non-user-based focus, your testing may focus more on benchmarking, comparing different algorithms or parameters, measuring performance or precision, etc. For any type of project, you can consider additional criteria where applicable, e.g., security, performance, accessibility, and computational efficiency. In the case of Cloud-based applications or services, one could also consider the cost implications (e.g., 'x' pence per query) and whether this has influenced the design and testing of the application.

Regardless of the project, you must describe the evaluation or testing of your system in your report, and this must include the following: a presentation of any relevant data; a discussion and analysis of the data; a discussion of the significant results and outcomes you have found. Ideally, you should consider any limitations in your evaluation and the extent to which your outcomes can be generalized to a wider ‘population’, or not.

Consider what you want to evaluate or test, and how you will achieve it. Develop the necessary evaluation plans / materials / methods, and make sure these are described in your report. Be mindful of ETHICS where required and make sure that the relevant Ethics documents are utilised, and it is clear where and how ethics has been adopted in your evaluation. Describe how your tests or evaluations were conducted. You can include the materials you have used in your appendices, e.g., test plans, evaluation checklists or tasks, copies of questionnaires used. Present and discuss the data in your report. You can include copies of the data in the Appendices too. Discuss the main outcome or findings from your evaluation / testing.

## Using subsections in your report

Remember, you can use subsections throughout your report to structure the content. This is often desirable to break up large expanses of text and to aid the reader too. Examples of subsections here, in the Evaluation section could be general such as Methodology or Results, and/or specific such as Usability, Performance, etc. Please remember to use the built-in styles for this that Word provides (Heading 1, Heading 2, etc.). This is necessary to ensure that your document is accessible.

### Here is a sub-subsection

You can use additional layers of hierarchy to progressively structure the content. In this case, if there was a subsection named Methodology, perhaps it could contain sub-subsections such as Participants, Tasks, Ethics, etc.

#### Be mindful of taking the structure too far

Whilst you can use as many hierarchies as you wish in structuring your content, there is usually a limit to what is useful in terms of readability. Aim to go no more than 3 layers deep in the hierarchy, if possible.

# Description of the final product

You should provide a clear description of what the final product looks like and what it does. You do not have to explore every minute detail of the system, you should attempt to convey the key, major areas of functionality. In some ways, you could consider this section to be a cut-down version of a user manual. Even in systems where there is no user interface, there may still be some general aspects that you can mention. However, if it is the case that this section of the report is just not relevant to your project, please just state that or omit this section.

When you are writing your report, you may find that the content of this section could overlap with earlier content in the report too, such as the implementation section. We want to avoid repetition in the report. At the same time, a degree of overlap is OK, bearing in mind that it is other people who are reading your report and they may benefit from a reminder, and a focused overview of what the final product looks like. As noted earlier, this section provides an overview of your finished product whereas earlier sections such as the implementation focus more on how you got to that point, i.e., the stages you went through, the decisions you made, and the problems you had to solve along the way.

# Appraisal

Provide a critical appraisal of the project. The question that I would pose to you here is as follows: if you were doing the whole project again, what would you do differently, what would you do the same, what advice would you give to others if they were doing the same project? Here you should reflect on the entirety of your project including your choice of technologies, your implementation decisions, and the project plan. With the benefit of hindsight, what are the lessons learned during the project and the evaluation of the final product and the process of its production (including a review of the plan and any deviations from it). Also consider what have been the most useful learning aspects for you.

NOTE: the appraisal section could potentially occur after the Summary and Conclusions below, or even as a sub-section within the Summary and Conclusions. See what works best for you and your advisor.

# Summary and Conclusions

Summarise the main points of what your project was and what the report has provided. Provide a summary. Describe the conclusions and outcomes that you have found.

# Future Work

What recommendations do you have for future work? Are there more features that need to be included? More testing? More evaluations? Are there follow-on projects or ideas that could be explored? Do you plan to do any more with the project yourself? Please discuss this here.

NOTE: this section could possibly appear as a sub-section within the preceding Summary and Conclusions.

## Acknowledgments

You can provide acknowledgements here to anyone who has been helpful in your project, or beyond. In some cases, the licensing of certain software products you have used may require you to acknowledge them here, e.g., in return for free use.

# References

[1] J. Bloggs and A. N. Jones, “Creating the perfect project”, International Conference on How to Get a First- Class Degree, Dundee, UK, 2020, pp. 1-6.

[2] M. Mouse "A comparison of automatic thesis generation algorithms", *IEEE Transactions on Document Generation*, Vol. 1, No. 1., pp. 900-919, 2112.

[3] I. N. Stein, D. Rac and D. Duck, *Transdimensional Monte Carlo Estimation of Honours Project Probability Mass Functions: Theory and Application*, Auchtermuchty University Press, U.K., 2053.

# Appendices

The appendices to your report will not appear here, they are submitted separately. However, you can provide a summary / bulleted list of what the appendices are here if you wish.