A-Level NEA Project

Contents

[Introduction 2](#_Toc115122101)

[Analysis 2](#_Toc115122102)

[Background To Project 2](#_Toc115122103)

[Research of Existing Solutions 3](#_Toc115122104)

[Casio FX-CG50 3](#_Toc115122105)

[Desmos Graphing Calculator 3](#_Toc115122106)

[Numworks Online Calculator Emulator 4](#_Toc115122107)

[Research Overview 5](#_Toc115122108)

[Interview 5](#_Toc115122109)

[Prototype Creation 6](#_Toc115122110)

[Display Libraries 6](#_Toc115122111)

[Maths Libraries 7](#_Toc115122112)

[Threading Libraries 7](#_Toc115122113)

[Final Objectives 8](#_Toc115122114)

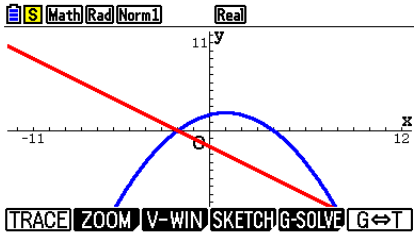
[References 9](#_Toc115122115)

# Introduction

My project will be based on the ability to display a graphing calculator on your computer. As I have been studying mathematics for over 12 years, I have become familiar with a wide range of solutions for calculators, whether they be hardware or software. However, as the complexity of the maths I have been working in has increased, I have ranged from using small pocket calculators to scientific ones. Also, last year when I started my Maths A-Level course, I was once again forced to upgrade to a gigantic graphing calculator. In my studies, having access to this huge calculator is more than beneficial, as it allows me to use it to execute commands quickly. However, the graphing application installed on the calculator is slow to run and difficult to pan and zoom, due to the slow 58 MHz speed of the processor on the calculator. So, my choice of designing my own solution to a graphing calculator came rather quickly. I feel that this project is complicated enough to challenge my understanding and collection of research, whilst remaining in the range of complexity suitable for an A-Level student.

# Analysis

## Background To Project

The aim of my project is to produce a fully interactive graphing calculator system that strives to solve some of the problems I have previously pointed out in the introduction, as well as some I will face while conducting research for my project, which is shown below. I will be basing my project on the graphing application for my calculator (see right), almost as if I am porting the application from the physical device to PCs. However, my project will be vastly more usable for use on a computer. The ways my project can be made to be better come from the increased set of features a computer can give. This includes but is not limited to:

* More complex manners of input (keyboard, mouse)
* CPU clock cycle speeds that are levels of magnitude faster than that of physical devices
* Larger screen resolution, allowing more data to be shown at once

Using this existing application for my calculator, I can create a solid set of basic functions that I would like my application to carry out:

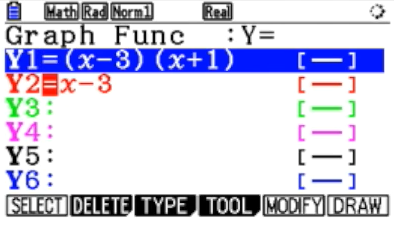
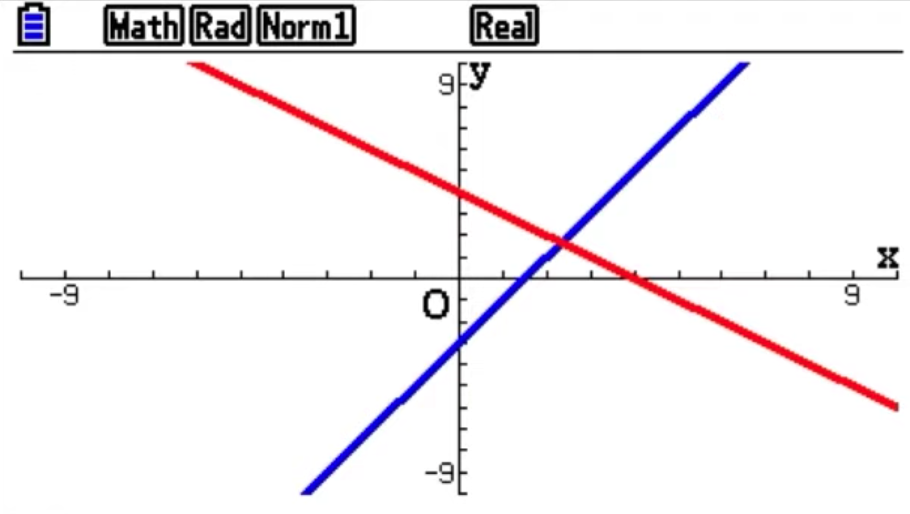
* A window that shows the axis and the plotted graph function
* A separate window that shows the UI and allows the user to change various parameters
* A subsection of the UI window that allows various variables to be calculated (like the intersection(s) of two graphs)

## Research of Existing Solutions

After browsing around for current solutions to the application, I have found three particular solutions that I would like to analyse further to gain a deeper understanding into what an optimal solution is. I have evaluated what I think are the best strengths of each solution, which I will strive to include in my application. On the other hand, I have also found what I believe are to be the worst weaknesses of each solution as well, these I will solve in my application. Here are my three investigations of each application.

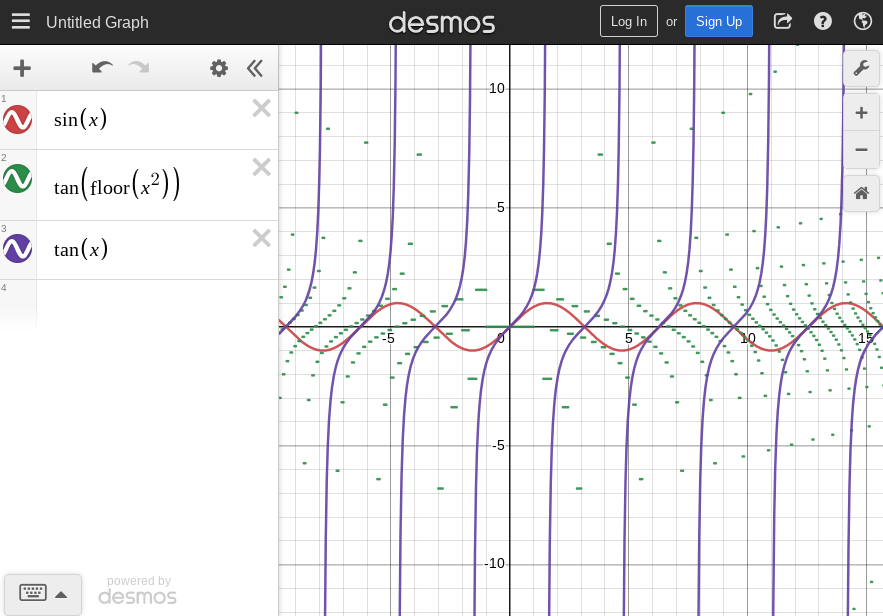
### Casio FX-CG50

The first application I thought to look at was my own graphing calculator: the Casio FX-CG50. This calculator has many functions, one of which being the ‘Graph’ application. It allows for equations to be entered that can be of many diverse types, like y=, x=, and r=, and has various functions including trigonometric, logarithmic, and hyperbolic functions - these are useful in GCSE and especially A-Level where more complex mathematical functions are necessary. It is a rather powerful tool when you have learned it, but it is very time-consuming and difficult to learn the full ins and outs of the machine - the user’s guide is over 620 pages long!

Another problem with the device is the size and resolution of the screen. The screen has a resolution of 216 x 384, which is workable, but the calculator commonly uses massive fonts that decrease the calculator’s usability. As well as this, it makes some of the applications awkward to use, namely the ‘Graph’ application, where panning and dragging the graph around is an often-repeated process. This is made worse by the fact that the calculator takes a solid second or two to calculate the result, which makes using the graphing application a time-consuming, slow process. Finally, this calculator comes with a hefty price-tag of £130, which is not ideal for anyone, let alone a schooling environment where the budgets cannot cover this much of a financial burden.

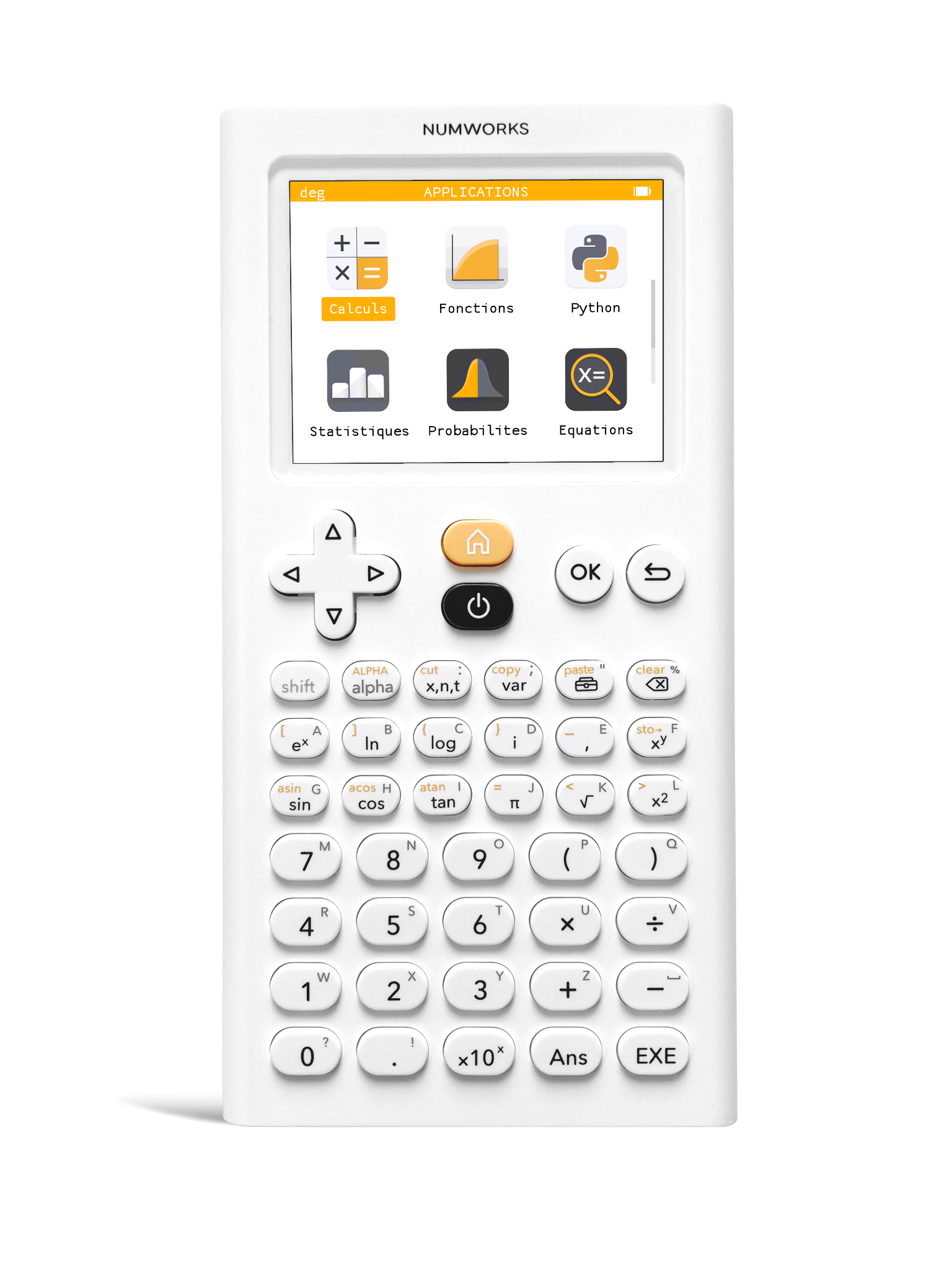
### Desmos Graphing Calculator

The second piece of research I conducted to do with the project was the Desmos online graphing calculator. **[1]** *(This is my notation for references, which can be found at the bottom of the document.)* Being online, this calculator is accessible for everyone and is an immensely powerful tool for showing these graphs on a set of axes. By simply typing in an equation into an entry box, the calculator is able to show the graph.

****The Desmos calculator solves the issues produced by the former Casio FX-CG50 calculator, such as price. This is completely free and completely accessible to all, provided you have an internet connection. This is the main issue with Desmos, that it is completely dependent on your connection to the internet. If you do not have a connection to the internet, then you cannot use this application in any manner, apart from the mobile version.

The mobile version is however rather difficult a lot of the time due to the lack of precision that you achieve with your fingers on a small 6” screen than that of a mouse on a bigger laptop or desktop. I feel that too much was crammed into the offline mobile version of Desmos as it is half the time, when you want to zoom in on a point, if you accidentally only zoom on one axis, only that axis will zoom in. That is a good feature, but for the majority of times it ends up making the graph look incorrect, as one axis is scaled more or less than the other. The keyboard input for this mobile version is so painfully small as well that I often find myself pressing the wrong button multiple times, then having to rewrite the entire equation.

### Numworks Online Calculator Emulator

Numworks is an American company that, like Casio, produces physical calculators for use in exams, especially to correspond with the American syllabus. Alongside the calculator, there is an online emulator **[2]** that can be used to imitate what the real hardware would perform as software on the computer. However, this is not online only: the .html file holding the emulator can simply be downloaded allowing for use of this system offline.

This is similar to a thing that Casio did, where you could download an emulator for their own Casio FX-CG50 calculator. This was, however, locked behind a requirement to input the licence key for your own physical calculator. Because I got my hands on my calculator through my school, and was just directly given it, I was unable to input the licence key on the packaging into the installer and therefore cannot install the software. Through looking at pictures and reviews of the software online, I can conclude that the emulation is rather poor in regard to its accuracy, and visual glitches and artefacts are plentiful when using this emulator.

Returning to the emulator from Numworks, the translation between inputs on the keyboard and inputs on the calculator are far from ideal. A select few buttons on the keyboard map directly to buttons of the calculator, but most of the buttons are required to be clicked using the mouse. This makes the ergonomics of this emulator far from efficient, as your hand needs to constantly move back and forth between the keyboard and the mouse.

Also, the user interface is designed for the calculator, so, once again, the ergonomics do not translate very well. To be honest, the whole operating system on the calculator is spread out so sparsely that there are probably many functions of the calculator that I have not even found in my research of it. Compared to the earlier Casio calculator, the font size is dialled right back, and screen size and resolution is increased, meaning that using the thing is no longer so zoomed in and so becomes more usable. But it seems that this new screen real estate has not been used, giving the calculator lots of room where something could be displayed but isn’t. This is not as big as an issue with the input translation, but an issue, nonetheless.

### Research Overview

These three examples of graphing calculators, whether physical devices or software on the internet, all have certain issues when it comes to their usability. It is all comes down to one or more of these three issues for each of these pieces of software:

1. The software is only accessible when unrelated criteria is met (like the use of the Internet)
2. The software uses UI that is too tightly compressed to fit in the screen
3. The software has bad input translation between the user and the interface

## Interview

To help develop the final objectives I need to gain a better understanding of the user requirements for the project, I conducted an interview with my Maths teacher Mr Smith. Here are some of the questions I asked him and alongside them the responses he gave.

**“How do you feel about the usage of these Casio graphing calculators in class?”**

**“**The Casio calculators are, by all means, powerful devices that can perform a multitude of tasks. However, I think that, in some ways, they have too much functionality, so much so that to the students it is exceedingly difficult to get the solution to a question. A lot of the time, my students ask me how to do certain things, especially in the graphing application, and a lot of my lesson time is wasted on showing them how to do things. This doesn’t surprise me, sometimes I can’t find something on it!”

**“Do you think the Desmos application provides a good interactive experience to plot graphs? How beneficial is it to your lessons?”**

“Desmos is a very haphazard application for me. At times, I can load it up and quickly show what a plotted graph looks like. I find its user interface kind of lacklustre but usable, and it can quickly show a graph. I also very much like that you can pan and zoom the image, to show how the graph looks when zoomed in or out or in a different position. My main issue with Desmos is that it requires the internet. My office/classroom is in a position where the internet is frequently dropping out, or it might be a dodgy piece of hardware in my laptop. Either way, a lot of the time I am unable to access Desmos and so as an application, half the time it doesn’t perform what I need.”

**“What would you want in an application that plots graphs?”**

“It is a necessity that it is usable offline, for the same reason I gave in your last question. It doesn’t help if the software is unavailable to me half the time. Also, a quick easy to use user interface would be beneficial to everyone.”

**“What do you feel should be at the forefront of a project like this?”**

“Usability. The amount of time I waste trying to find out how I press or do something in one of these applications is not to be understated. I am not particularly good with computers in the first place, so I find it difficult to do some things in the software. It should have an easy-to-use GUI with self-intuitivity at the forefront.”

## Prototype Creation

### Display Libraries

When creating a project, it is important to work out what programming language you are going to use and which libraries you intend to make use of. For this project, I have decided that I want to use Python to strengthen my knowledge of the language. As for the libraries side, I was initially intending to draw the screen exclusively using the *Tkinter* framework, which neatly includes a separate module called *Turtle*. I found using stackoverflow.com **[3]** that these two libraries can work together to create a very nice-looking window-in-window look that is rather aesthetically pleasing. However, when writing some prototype code, it really showed how slow Turtle is to run. This is unsurprising, since this is not what the library was designed to do.

After I conducted further research, I found out that there is another module called *Pygame* **[4]** that is much more optimised for speed. This is a huge set of libraries that are designed for creating video games, however it is not exclusive to this category of software. Here are a few reasons why Pygame makes more sense for this project than Turtle:

1. Pygame uses optimised C and assembly code for its core functions. Code written in C is often 10-20 times faster than Python, and assembly code can often be one hundred times faster.
2. Pygame comes with many operating systems and is truly portable between several types of computers.
3. Pygame is very programmer-central – meaning you call Pygame functions. This gives greater control over the program to the programmer.
4. Pygame has a remarkably simple input system, which can be implemented in code easily

Because of these benefits, I have decided to use Pygame for my main window where the axis and plotted graph are displayed and use Tkinter alongside it for the main UI where equations are inputted, in a separate window.

### Maths Libraries

For a graphing calculator, obviously there will need to be maths involved. For basic functions including trigonometry and logarithms, the basic built-in library for maths in Python will do the job. However, for more complex functions, I will be making use of the *NumPy* **[5]** module. This aims to enable numerical computing and adds data structures like matrices. It also allows users to perform arithmetical operations on such data structures. As well as NumPy, I will use *SymPy* **[6]** for its implementation of symbolic mathematics into Python. SymPy’s features include solving polynomials, calculus and solving generic equations.

### Threading Libraries

Threading in computers means when the computer runs multiple pieces of code simultaneously. This will be a must-have in my project due to the complexity and vast processing time required for the mathematics involved in the project. This can be done in Python by using the built-in library *threading* **[7]**, but there is one fatal flaw with this library. This glaring oversight is that is does not really carry out threading, it just shares a single processor core’s processing time between the current threads, instead of making use of the dual-core, quad-core and even octa-core processors of the 21st century.

This is where I found out about another built-in library *multiprocessing* **[8]**. This is a package that supports spawning processes using an API, and it allows the programmer to fully leverage multiple processor cores on a given machine by spreading out processes between cores, instead of just allocating processing time to a singular core. This accelerates the speed of the program by however many cores your computer has, 2x for dual-core, 4x for quad-core etc.

The issue with the multiprocessing library is that it is extremely difficult to use. Memory can only be shared between the main thread and the subprocess by use of a queue, which can be awkward to use. Also, all data passed into the process must be serialised, which can take up to a few seconds. This means there is a few seconds of delay before the process is started, which leads to lag. Therefore, one process should be created and looped over and over without starting any new processes. I wish to implement a one-queue-in one-queue-out system for passing data between threads.

Without the usage of these threading systems, the program would run at just a few frames per second, which is not ideal for this type of program. Implementing threading alleviates the processing time required in the main thread, allowing for much higher FPS. At the moment I plan to cap the FPS at 30, but if I can optimise my code enough, I will perhaps be able to run my program at 60 FPS.

## Final Objectives

To ensure I stay on track throughout my project and that the program meets the end of the intended audience, I have produced a set of key objectives, which will be met throughout the production of the project.

1. Implement a display window in Pygame
   1. Draw a set of axes on the graph
      1. Make these scale with the zoom value of the graph
   2. Allow the user to pan and zoom the graph
      1. Panning and zooming with the mouse
      2. Panning and zooming with the keyboard
   3. Inputted equations are drawn onto the screen
      1. They are drawn in the correct position
      2. They are updated every few frames
   4. Make sure the display window is resizable and adjusts correctly when resized
   5. Draw a second UI on top of the display window
      1. Make sure it writes graph offset and zoom
2. Implement a user-interface in a separate window with Tkinter
   1. Input boxes for equations to be written in
      1. When changed, these update the drawn graphs on the screen instantly
   2. Buttons to directly change pan and zoom values to inputted value
      1. Insert another button beside to update to inputted values when clicked
3. Implement section of buttons that can be used to calculate values relating to the current equations
   1. Make sure all the values to these pieces of data appear in a small *message box*
   2. Add a button to calculate the y-intercept of the currently highlighted graph
   3. Add a button to calculate the root of the currently highlighted graph
   4. Add a button to find the intersection points of two graphs
   5. Add a button to find the exact value at an exact point on a graph
      1. This allows a X or Y coordinate to be inputted and then the other of the two is calculated and displayed in a window
4. Implement a system in which multiple processes can be run simultaneously using the multiprocessing library
   1. Processes are assigned two queues when started, an “in queue” and an “out queue”
   2. Zoom and offset values are passed into processes through the use of the “in queue”
   3. An array of numbers is passed from the process back to the main thread through the use of the “out queue”

# References

|  |  |
| --- | --- |
| [1] | <https://www.desmos.com/calculator> |
| [2] | <https://www.numworks.com/simulator/> |
| [3] | <https://stackoverflow.com/questions/44634947/how-to-set-a-turtle-to-a-turtle-screen/44639041#44639041> |
| [4] | <https://www.pygame.org/wiki/about> |
| [5] | <https://numpy.org/> |
| [6] | <https://www.sympy.org/en/index.html> |
| [7] | <https://docs.python.org/3/library/threading.html> |
| [8] | <https://docs.python.org/3/library/multiprocessing.html> |