Fractal Visualisation Application

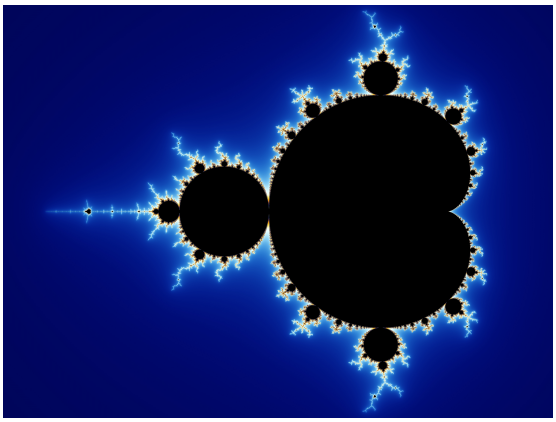
Contents

Analysis

In this project, I intend to create a simple but well-featured graphical interface for exploring fractals, a mathematical phenomenon arising from repeated iterations of mathematical equations. The project will therefore consist of two main parts, the backend (responsible for calculating and producing the fractal itself, and usable from a terminal for more advanced users) and the frontend (responsible for displaying and allowing interaction with the user in real-time, graphical and intended to be easy to use).

Background to the Problem

Fractals



Render of the Mandelbrot Set (created by Wolfgang Beyer, via Wikipedia)

Fractals are a mathematical phenomenon which arise from repeated iteration of a recursive mathematical equation, the most well-known being the Mandelbrot set. The equation responsible for producing it is $z = (z^2) + c$. A fractal is 'rendered' into an image by taking each pixel, calculating a position for it in the complex number space (the horizontal axis becomes the real coordinate, vertical axis becomes the imaginary coordinate), and assigning that complex number to both variables z and c initially. Then, the equation is iterated repeatedly, with c staying constant between iterations and z being recalculated each time. Iteration continues until z tends to infinity (usually defined as having a modulus greater than 2, after which it is guaranteed to tend to infinity), or until some defined limit is reached, at which point the number of iterations completed is counted or reported. The number of iterations performed then becomes the 'value' assigned to that point in the complex number space, and be used to select a colour for a particular pixel.

Fractals are interesting primarily due to their property of self-similarity: no matter how closely you zoom in or how high a resolution image is rendered (i.e. how fine the arbitrary 'pixels' are) there will always be an infinite depth of detail. The complex high-level shapes, such as the recognisable form of the Mandelbrot set, are repeated (although not necessarily identically) in smaller and smaller instances, with many other features such as spirals, ‘cracks’, and more. Thus, it is impossible to represent fractals using vector images, due to the fact that they have an infinite level of edge detail, and effectively an infinite circumference therefore.

Their self-similarity and detail properties has led them to be compared to (and indeed modelled against) natural shapes, such as the growth of fern leaves and coastlines (the coastline paradox is an observation that as a coastline is measured more and more precisely, its circumference seems to continue to increase towards infinity, known as having a fractal dimension). Thus, studying mathematics in this way can be very useful in understanding natural phenomena and real-world

problems. This also makes fractals a useful demonstration for mathematics students (or anyone in fact) as discussed further below.

Mathematics

Mathematics is often seen as an uninteresting topic or field, at least for anyone who doesn't already have an interest in it. It tends to be defined by equations and numeracy which many people find boring. However, fractals are an example of how mathematical ideas can apply not only to modelling nature and drawing interesting parallels with the real world, but also in producing aesthetically pleasing and beautiful images which everyone can enjoy and appreciate, perhaps even as an art form. Particularly, their use as an illustration of the applicability of mathematics in a classroom can make the subject more engaging, especially to school students.

Thus, this project aims to, as well as providing a useful fractal visualisation tool, be primarily a teaching aid which can give students the opportunity to experiment with fractals and learn about why they occur at the same time, in an engaging fashion.

The Problem

This brings into focus the problem which the project will address, namely the engagement of students or non-mathematical people with maths in general. Currently, tools for visualising fractals are limited, many are either demos with expensive full versions, outdated, slow, or difficult to use:

- Xaos - a free, interactive piece of software, but has a limited functionality and lacks the functionality to input custom equations and display them as fractals
- FRAX - cheap, but only runs on iOS devices and is thus limited in its processing and usability, making it difficult to render with more computationally expensive parameters and very hard to demonstrate to others
- Fractal to Desktop - less fully featured, simpler to use, but only supports Windows (might be inaccessible for some students who do not have access to a Windows-hosting device)
- JWildfire - although very powerful, also quite complex and has a steep learning curve to it
- Ultra Fractal - incredibly powerful and fully featured, but allows only a 30-day free trial before costing £25-80
- Apophysis - supports a limited range of fractals but lacks help/instructions for users

The user of my application is likely to be one of three main categories:

- A teacher looking to engage and excite students and make maths classes more practical (particularly for teaching topics such as complex numbers or iterative/numerical methods)
- A student learning maths interested in exploring mathematical phenomena or putting them to the test
- Ordinary, non-mathematical people looking to experiment in abstract design or maths

However, my primary **client** for the project will be a teacher from the mathematics department at my school. I interviewed them on their thoughts about the problem and basic requirements for a solution.

“

This tool would be very useful as a form of enrichment to the maths syllabus. I might expect to use it towards the end of term, or early on in the school to show students how exciting and beautiful maths can be, especially given the intriguing nature of fractals. This would also be an interesting extension exercise for sixth form students to investigate applications of complex numbers.

A solution would need to be quick to set up and responsive, to keep students engaged. It should also be visually appealing. There should be the ability to see how the images are generated for sixth form students interested in maths and computing.

”

Solution Requirements

Following my research and a brief discussion with my user, the following requirements are apparent. High level objectives are identified with a '1-xx', low level objectives with '2-xx'.

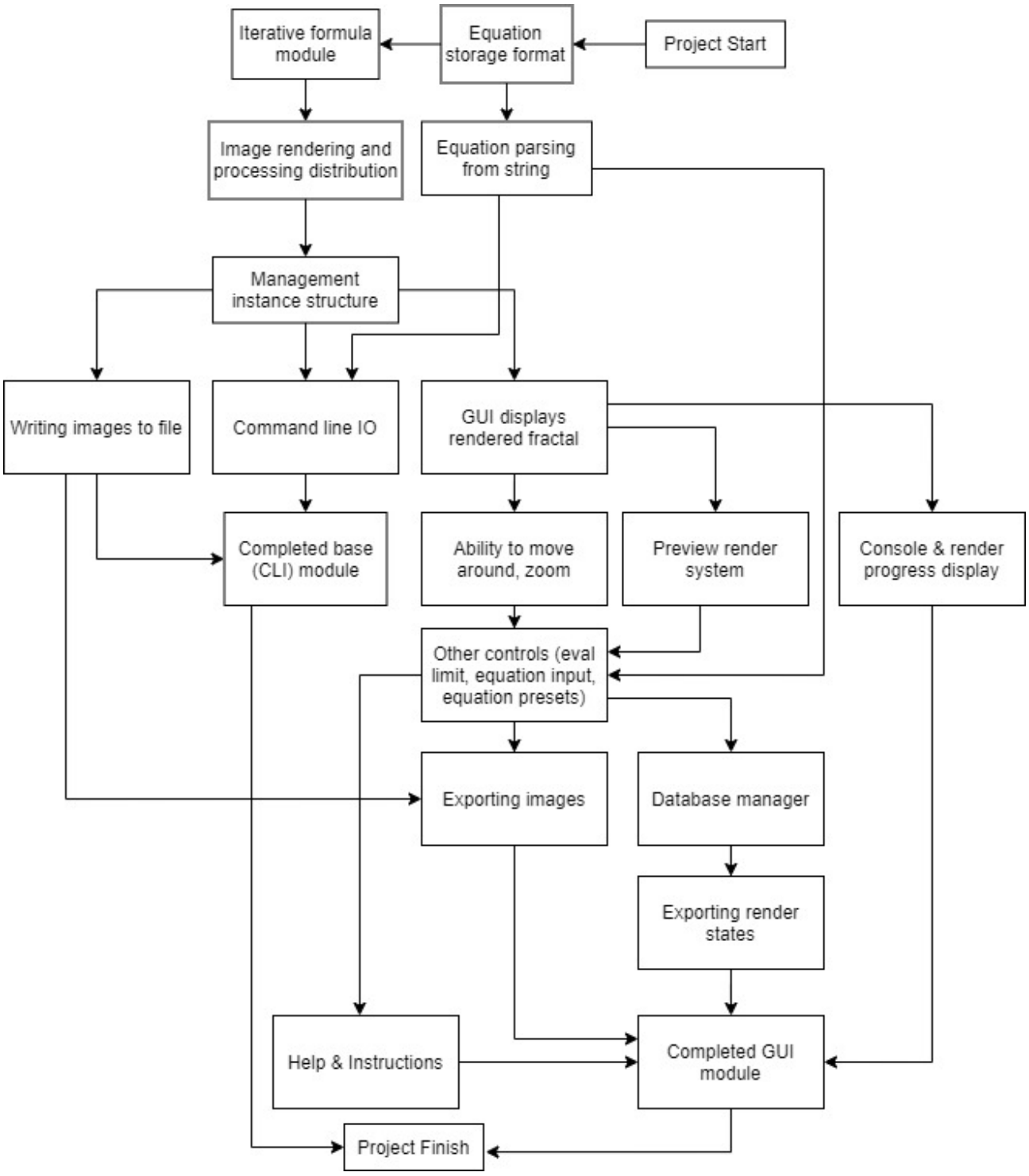
1-01: The program must be able to produce a rendered fractal at any resolution requested.
1-02: The program must be able to display the output to the user
1-03: The user must be able to move, navigate around, and zoom in and out of the fractal freely
1-04: The user should be able to change the equation being used
    a) The user should be able to input their own iterative equation (i.e. in place of `(z^2)+c`)
    b) The user should be able to select an equation from a list of built-in presets
1-05: The user should be able to save rendered images to a file
1-06: The user should be able to save configured states (i.e. a particular equation with a particular offset, zoom, etc) and restore them
1-07: The program should be able to generate a full screen rendered fractal in under 30 seconds
1-08: The interface must be pleasant to use and intuitive, such that a maths teacher can use it easily without confusion
1-09: The program should offer or come with instructions on how to get started and use each of its features
1-10: The user must be able to change computation and display parameters, such as the iteration limit and colour scheme
1-11: The program must be able to run on both Windows and MacOS machines

2-01: The program should present some kind of preview render which has more minimal parameters (lower resolution and iteration limit) to allow the user to navigate quickly (without having to wait for a full resolution render in between every navigation event)
2-02: The user should be able to navigate using both interface buttons and keyboard keys
2-03: The program should be runnable both as a GUI or as a CLI tool for more advanced users
2-04: The program must be able to parse mathematical expressions from a string including involving imaginary parts and evaluate them efficiently at runtime, including the following operations:
    a) brackets
    b) implicit multiplication
    c) addition
    d) subtraction
    e) multiplication

    f) division
    g) indexing (powers)
2-05: The program must be able to utilise multithreading in order to speed up computation
2-06: The program must be able to store and load render configuration states in a database structure when the user clicks the relevant button
    a) When storing, the program should present a dialog which allows the user to save the render state
    b) The user should be able to name their render state
    c) The database should be stored in an appropriate location on disk
    d) When loading, the program should present the user with a dialog which allows the user to select a saved render state to load
2-07: The GUI must be able to resize and adapt its render resolution to match that of the screen for optimal user experience
2-08: The program should have an instructions area which is easy to find from the main window.
    a) The instructions should explain how to render an image
    b) The instructions should explain how to save and load render states
    c) The instructions should explain how to input custom equations, and also should describe the presets
    d) The instructions should explain the meaning of the colour scheme, zoom, offset and iteration limit parameters
    e) All of these should be explained in a way which a maths teacher or student can understand
2-09: The program should have a sensible requirement of memory to run (less than 500MiB) at any one time, which should remain consistent even if the program is open for a large period of time (dependent on the resolution being rendered at)
2-10: The program should display its rendering progress (if there is a render currently happening) and its current status, to the user
2-11: The program should lock controls such as iteration limit, offset and zoom whilst a render is occurring

Project Development Plan

Below is a flow diagram showing the order in which I plan to carry out development tasks for this project. The order and requirements for each stage are flexible, as they may change due to further design improvement as I design and develop the project.



Necessary Libraries

The solution to the problem necessitates the use of a graphical user interface. For this, a GUI library is necessary in order to produce a consistent experience across platforms.

A wide range of solutions are available for this. My criteria for a package were as follows:
- free to redistribute
- free to use for commercial purposes
- configurable inside another project
- compatible with the target language, in this case C++
- cross-platform
- providing features such as displaying images, buttons, text

The potential solutions which I identified included Qt, GTK+, wxWidgets, FLTK, BoostUI, and gtkmm. However, many of these did not fit my criteria fully, so I finally chose to use 'raylib', a standalone header-only graphics library which meets all the requirements I set out with.

The C++ standard library fortunately has a built-in complex number handling library. However, during my analysis phase I initially experimented with configuring and using multiple-precision complex number libraries. Multiple-precision arithmetic, or arbitrary precision arithmetic, are methods of performing arithmetic which is not limited by the word size of the host system. The complex numbers involved in generating fractals must be very precise, so I considered using arbitrary precision arithmetic libraries such as:
- MPFR (Multiple Precision Floating-point Reliable) - provides extensible floating point data types which can be sized to any necessary precision
- GMP (GNU Multiple Precision) - a similar library but which determines to provide better performance
- MPC (Multiple Precision Complex) - library which builds off of GMP to provide complex number arithmetic at arbitrary precision
- Boost multiple precision complex - the Boost implementation of the same functionality

However, I encountered a number of problems with all or most of these during my experimentation:
1. They caused a huge performance hit when benchmarked compared with the C++ standard complex number library, on the scale of orders of magnitude slower (which is to be expected with any arbitrary precision arithmetic realistically), which would cause significant problems due to the number of calculations necessary in this project and the intended real-time nature of it.
2. They required the target machines to have libraries installed in order to use the project and application, which also would require configuring host machines (going against the aim of the project to provide a simple, standalone, easy to install and use application)

As a result I decided to stick with the C++ built-in complex number library, the only trade-off being the limited maximum precision of calculations as a result. This has the effect of effectively limiting the resolution or zoom depth possible with the application, as at a certain combination of resolution and zoom, pixels will no longer be able to be uniquely identifiable and their complex coordinates will be rounded to the nearest floating-point number which can be stored. In order to minimise this, in my application I will aim to use the highest precision float possible (‘long double’ in C++).

Design

High Level Design

As mentioned in the analysis section, I decided to allow both command line and GUI operation of my application. As such, my design can be effectively split into two main parts:

- the fractal generator (rendering environment)
- the GUI

The fractal generator is the core of the application which does the actual processing and creation of the rendered image. It will be accessible from both command line and GUI modes, whereby either mode can create a rendering environment with the necessary parameters (zoom, offset, equation, etc) which can be used for rendering universally.

In CLI mode, this can be done by parsing command line arguments and passing them into the rendering environment as render parameters.

In GUI mode, this can be done instead by simply reading out values from the interface which the user can manipulate (e.g. the user presses a button which zooms in and this command is processed by changing a parameter in the rendering environment).

The GUI is effectively a separate module which provides the graphical interface to the user and simply modifies and calls into the rendering environment provided by the core module and displays the results on the screen, as well as other features such as saving images and accessing the database of render parameters.

The rendering environment will be **encapsulated** in a class which can be instantiated to create an independent, contained environment for generating fractals. It will contain class variables which can be assigned or retrieved via getters and setters. It will also provide functionality to render fractals and to parse textual equations (see below), **hiding all the code** necessary to do this from anything which utilises the rendering environment (e.g. the GUI module).

The GUI module will also be **encapsulated** in a class, which is host to the entire graphical system. It will be responsible for creating and managing the main application window, handling button presses and key events, and will keep track of the rendering environments (two will be used, one for the low resolution preview render, the other for the main, full-resolution render). The GUI class will contain various methods for handling presses for each button, navigation events like moving the viewport, as well as the main-loop function which will manage drawing the interface each frame. The only externally accessible method will be the GUI class's main function, which will perform setup such as intialising the class, creating the GUI and starting the graphics main-loop, from which point the GUI will occupy the main program thread until the user closes the window, when the program will exit.

The ‘int main(int, char**)’ function, which is called when the program starts, will contain code to handle either a console start or a GUI start. For a console start, it will parse the command line arguments, configure a rendering environment, perform the render, save the result to a file, and exit. For a GUI start, instead it will simply construct a GUI class and transfer control to the graphics main method.

Additional Modules

Equation Parser

Other smaller subsystems will include the equation-parsing module, a set of static, **stateless** functions to convert a mathematical expression (e.g. "(z^2)+c", which represents the Mandelbrot set fractal) stored as a string into a form which can be evaluated rapidly at runtime. This module will define a number of data types to represent tokens which make up the parts of the equation.

More information about the equation storage and parsing is below, where the algorithm used and data structures required are described. In order to hide as much code as possible from other modules, only the outer-most of these functions will be accessible to the rest of the program, which will simply be called to perform the entire parsing and conversion operation and will return an equation object (see below).

Database

Another element of the application is the ability to save details of configuration states (effectively saving the configuration of the rendering environment, along with a name), via a simple **two-table database** stored in CSV format. This will have its own standalone module, with a class which can be used to manage, update, remove, or create records and handle saving and loading them automatically, again **encapsulating** the database system. This will involve two other data types, representing a record from each table. One table will contain configuration profiles, the other will contain the details of users. The configuration profile table will include fields for:

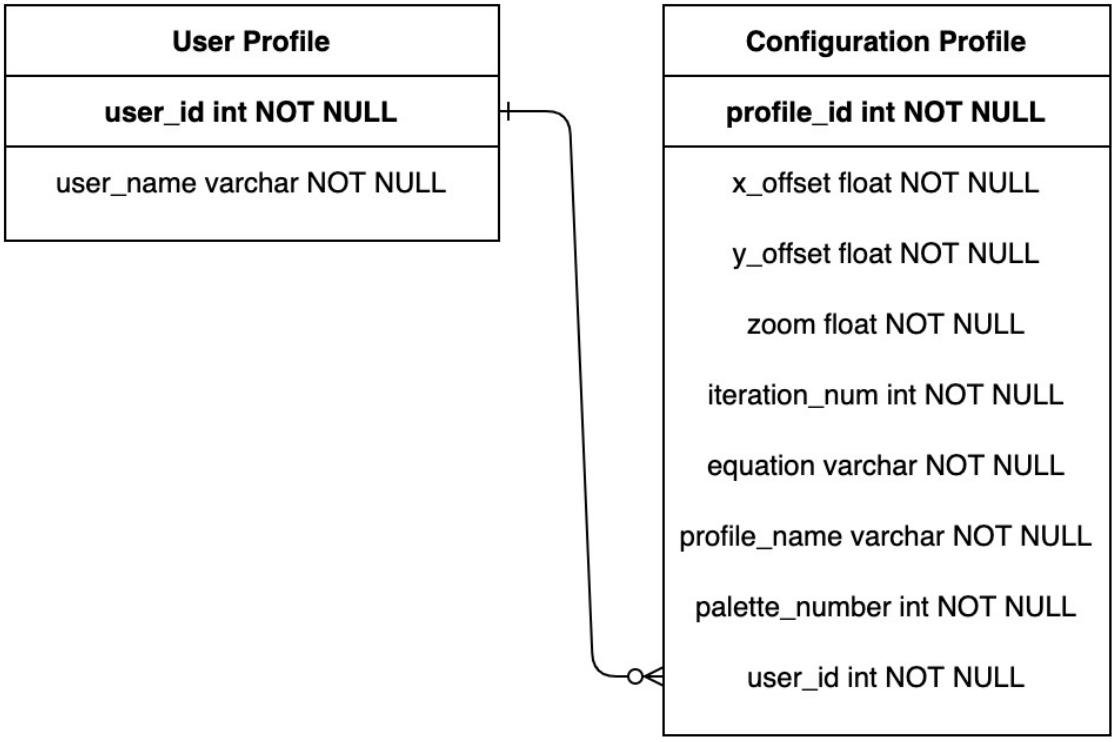
- Record ID (primary key)
- Config Profile Name
- Fractal X Offset
- Fractal Y Offset
- Fractal Zoom
- Fractal Equation
- Fractal Iteration Limit
- Palette Number
- User ID (foreign primary key of user profile table)

Meanwhile, the user profile table will contain fields for:

- User ID (primary key)
- User Name

The relationship between the configuration profile table and the user profile table will be many-to-one, since many configuration profiles can reference a single user profile (but only one user profile can be referenced by a particular configuration profile).

All of this is illustrated by the **entity-relationship** diagram below:



Evaluator

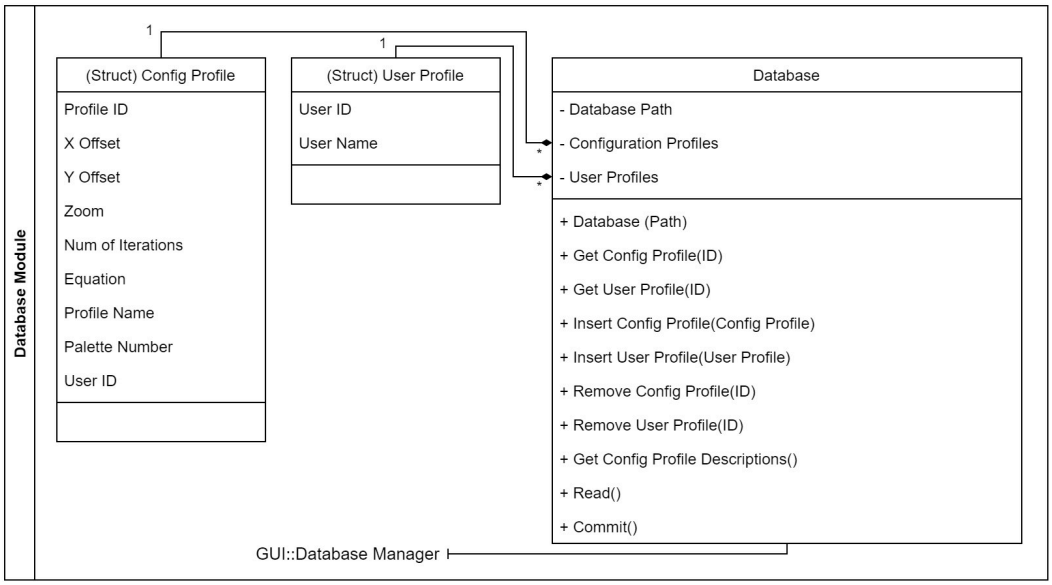
The final module will be the evaluator, which contains the code to handle and evaluate equations parsed by the equation parser. It will **hide the code** to actually evaluate values for any given complex coordinate given, providing functionality to do this which the rendering environment can call for every pixel which needs evaluating. In short, a function will substitute values for variables such as z and c into the stored equation and calculate the result in the form of a complex number. A second function will repeatedly perform this 'single iteration' as described in the analysis section (where the calculation is performed iteratively until the z value of the equation tends to infinity or until the specified iteration limit is reached), and an integer representing the number of iterations performed will be returned (which can then be used to colour each pixel).

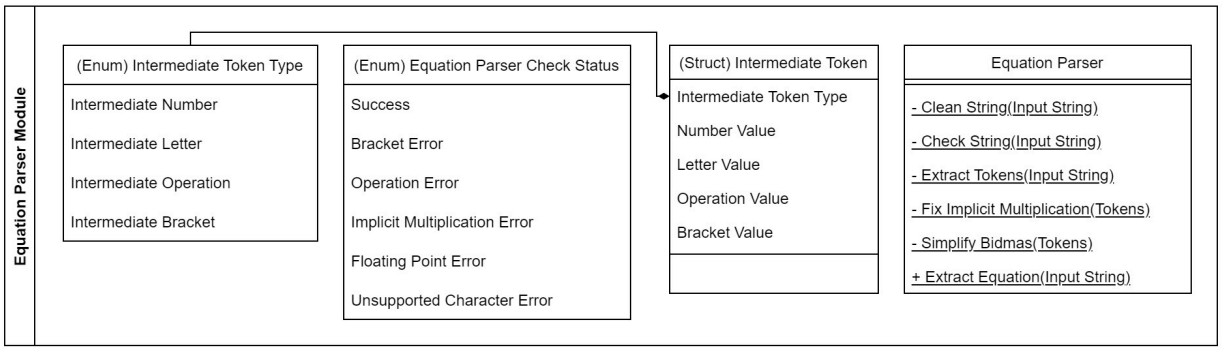
This will also be wrapped in a class to **encapsulate** the evaluation process as tightly as possible.

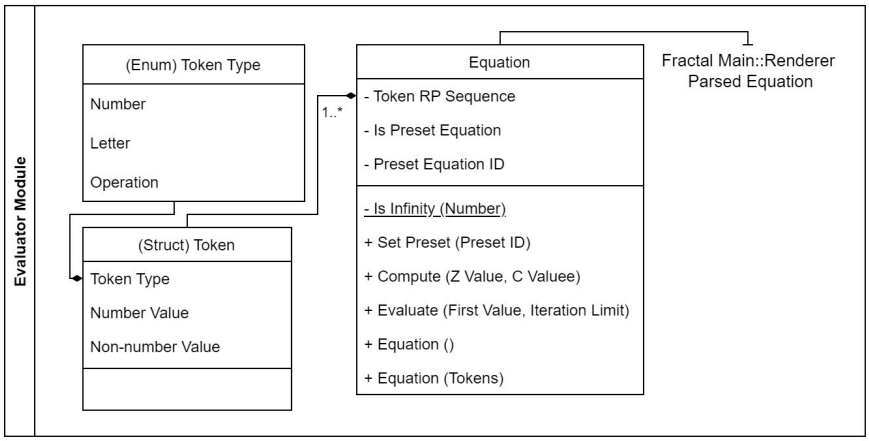
There will also likely be a utilities module which contains miscellaneous functions which multiple other modules might be using, such as a delay/wait procedure, fetching a system path, or retrieving information about an equation preset.

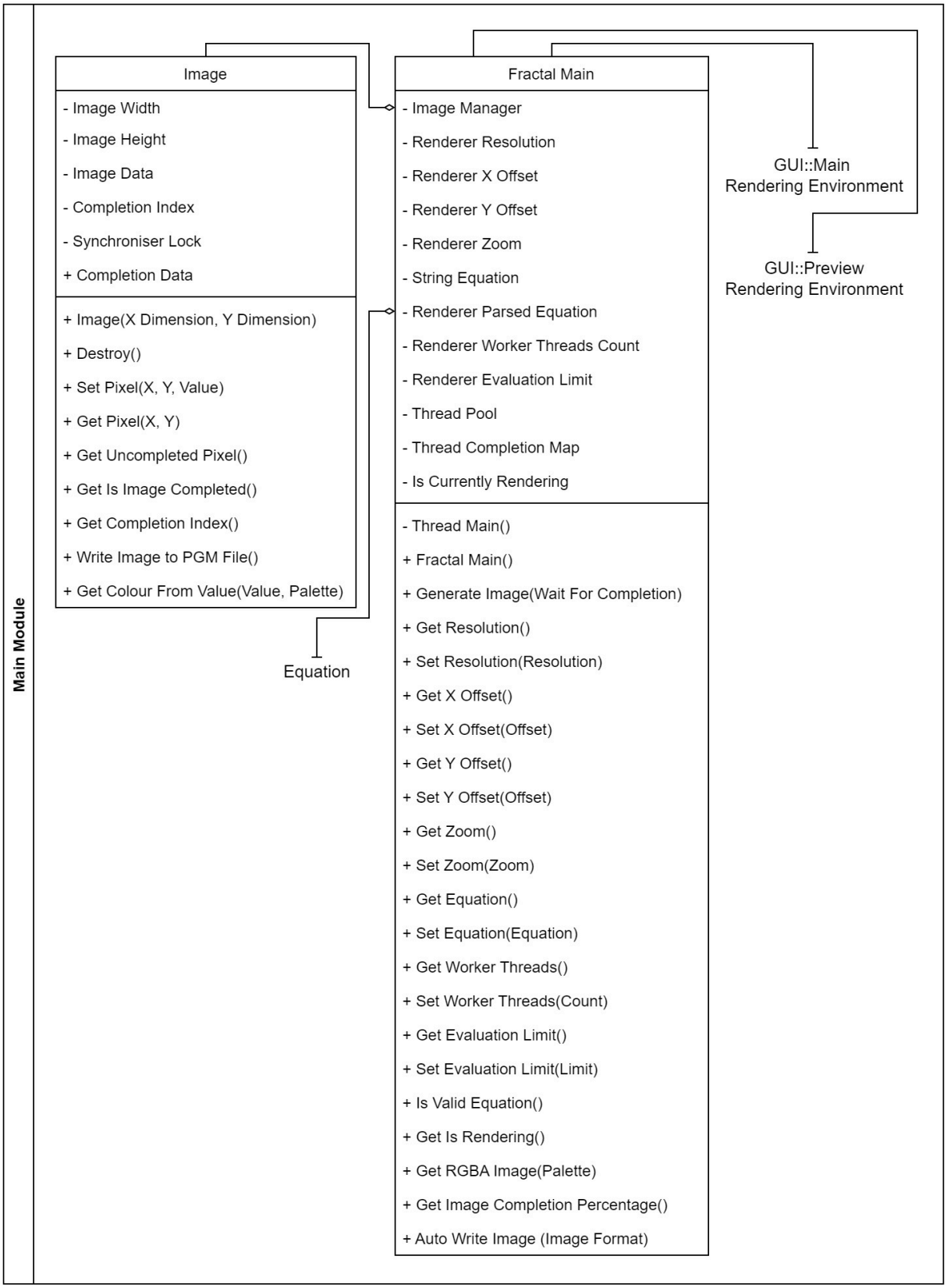
Class Diagram

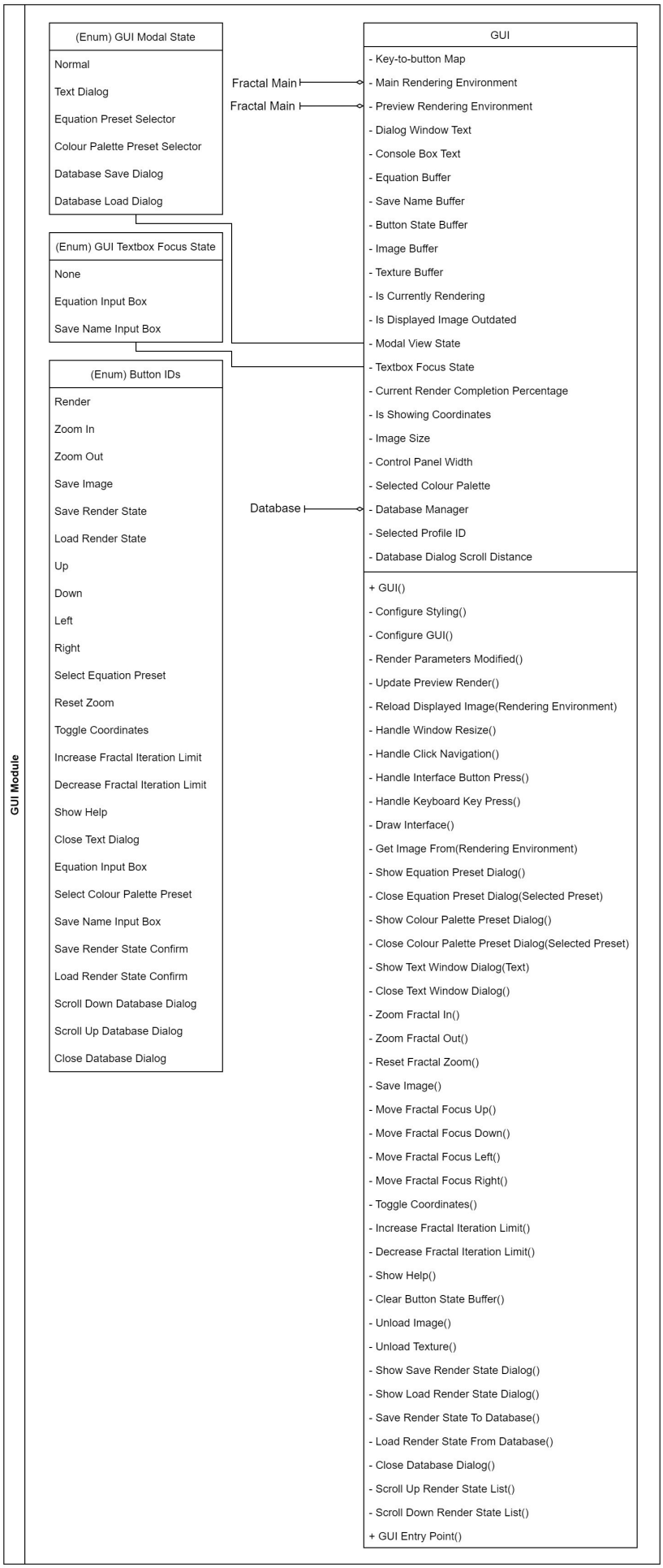
The following pages show comprehensive conceptual class diagrams, which have been separated into modules to make them easier to read. Due to the number of classes, structures and enumerations, the diagram has been split over multiple pages. Connections which split over page borders are shown with a label (as seen with ‘GUI::Database Manager’ which relates to the ‘Database Manager’ field on the ‘GUI’ class).



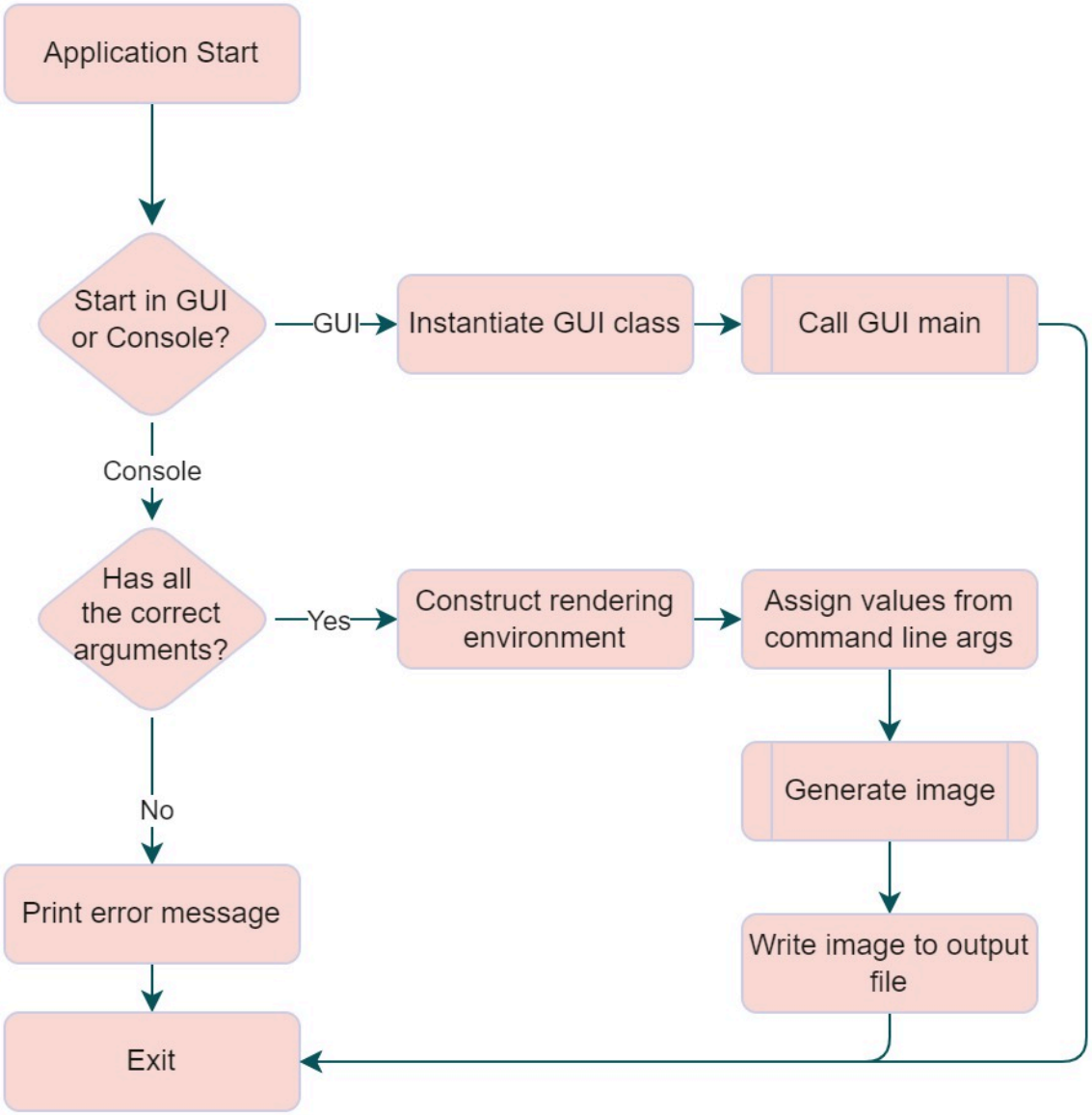


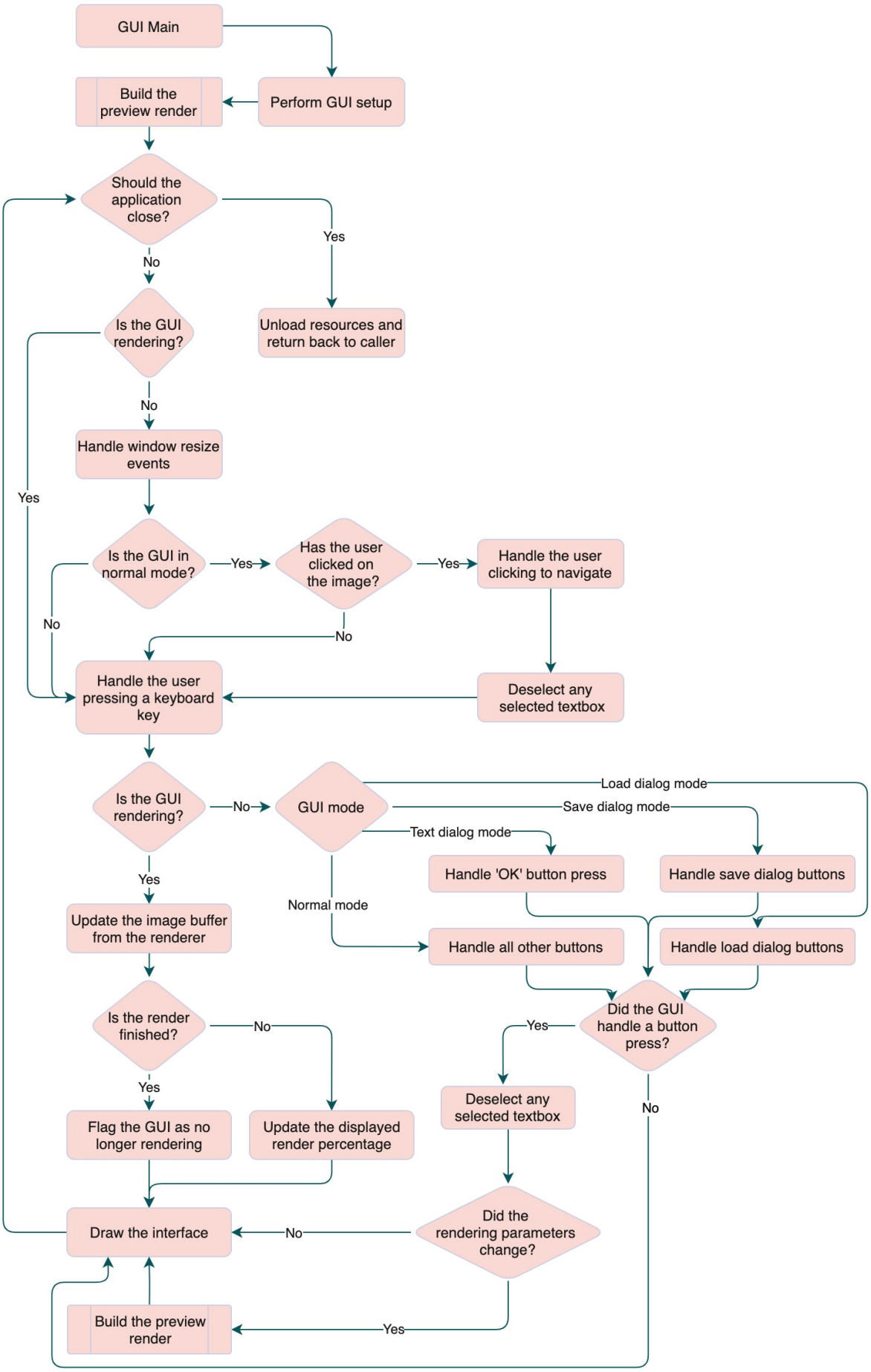


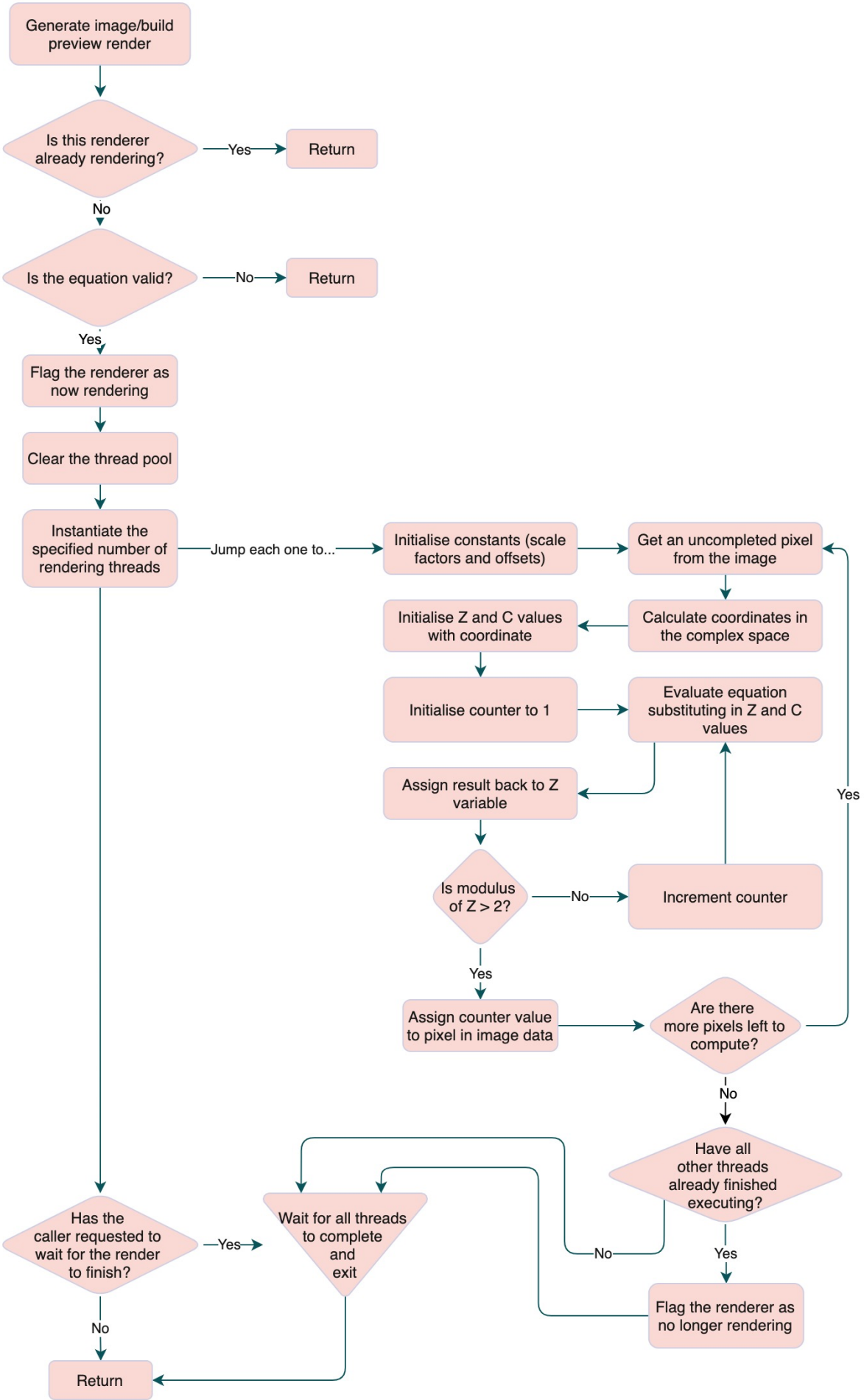

Main Module
Image
- Image Width
- Image Height
- Image Data
- Completion Index
- Synchroniser Lock
+ Completion Data
+ Image(X Dimension, Y Dimension)
+ Destroy()
+ Set Pixel(X, Y, Value)
+ Get Pixel(X, Y)
+ Get Uncompleted Pixel()
+ Get Is Image Completed()
+ Get Completion Index()
+ Write Image to PGM File()
+ Get Colour From Value(Value, Palette)
Equation
Fractal Main
- Image Manager
- Renderer Resolution
- Renderer X Offset
- Renderer Y Offset
- Renderer Zoom
- String Equation
- Renderer Parsed Equation
- Renderer Worker Threads Count
- Renderer Evaluation Limit
- Thread Pool
- Thread Completion Map
- Is Currently Rendering
- Thread Main()
+ Fractal Main()
+ Generate Image(Wait For Completion)
+ Get Resolution()
+ Set Resolution(Resolution)
+ Get X Offset()
+ Set X Offset(Offset)
+ Get Y Offset()
+ Set Y Offset(Offset)
+ Get Zoom()
+ Set Zoom(Zoom)
+ Get Equation()
+ Set Equation(Equation)
+ Get Worker Threads()
+ Set Worker Threads(Count)
+ Get Evaluation Limit()
+ Set Evaluation Limit(Limit)
+ Is Valid Equation()
+ Get Is Rendering()
+ Get RGBA Image(Palette)
+ Get Image Completion Percentage()
+ Auto Write Image (Image Format)
GUI::Main
Rendering Environment
GUI::Preview
Rendering Environment


GUI Module
(Enum) GUI Modal State
Normal
Text Dialog
Equation Preset Selector
Colour Palette Preset Selector
Database Save Dialog
Database Load Dialog
(Enum) GUI Textbox Focus State
None
Equation Input Box
Save Name Input Box
(Enum) Button IDs
Render
Zoom In
Zoom Out
Save Image
Save Render State
Load Render State
Up
Down
Left
Right
Select Equation Preset
Reset Zoom
Toggle Coordinates
Increase Fractal Iteration Limit
Decrease Fractal Iteration Limit
Show Help
Close Text Dialog
Equation Input Box
Select Colour Palette Preset
Save Name Input Box
Save Render State Confirm
Load Render State Confirm
Scroll Down Database Dialog
Scroll Up Database Dialog
Close Database Dialog
Fractal Main
Fractal Main
Database
GUI
- Key-to-button Map
- Main Rendering Environment
- Preview Rendering Environment
- Dialog Window Text
- Console Box Text
- Equation Buffer
- Save Name Buffer
- Button State Buffer
- Image Buffer
- Texture Buffer
- Is Currently Rendering
- Is Displayed Image Outdated
- Modal View State
- Textbox Focus State
- Current Render Completion Percentage
- Is Showing Coordinates
- Image Size
- Control Panel Width
- Selected Colour Palette
- Database Manager
- Selected Profile ID
- Database Dialog Scroll Distance
+ GUI()
- Configure Styling()
- Configure GUI()
- Render Parameters Modified()
- Update Preview Render()
- Reload Displayed Image(Rendering Environment)
- Handle Window Resize()
- Handle Click Navigation()
- Handle Interface Button Press()
- Handle Keyboard Key Press()
- Draw Interface()
- Get Image From(Rendering Environment)
- Show Equation Preset Dialog()
- Close Equation Preset Dialog(Selected Preset)
- Show Colour Palette Preset Dialog()
- Close Colour Palette Preset Dialog(Selected Preset)
- Show Text Window Dialog(Text)
- Close Text Window Dialog()
- Zoom Fractal In()
- Zoom Fractal Out()
- Reset Fractal Zoom()
- Save Image()
- Move Fractal Focus Up()
- Move Fractal Focus Down()
- Move Fractal Focus Left()
- Move Fractal Focus Right()
- Toggle Coordinates()
- Increase Fractal Iteration Limit()
- Decrease Fractal Iteration Limit()
- Show Help()
- Clear Button State Buffer()
- Unload Image()
- Unload Texture()
- Show Save Render State Dialog()
- Show Load Render State Dialog()
- Save Render State To Database()
- Load Render State From Database()
- Close Database Dialog()
- Scroll Up Render State List()
- Scroll Down Render State List()
+ GUI Entry Point()

Program Flow Diagram

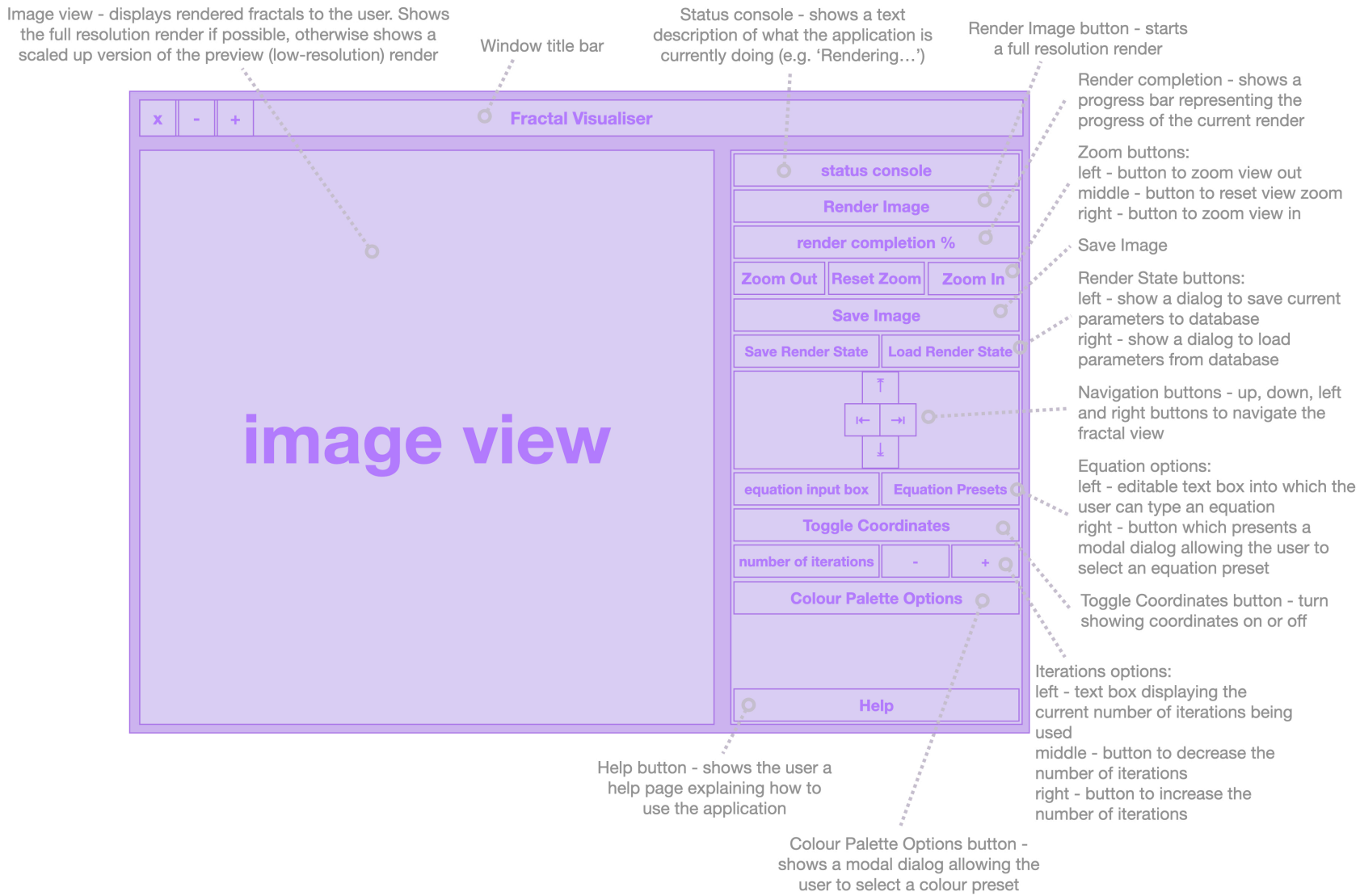



GUI Main
Perform GUI setup
Build the preview render
Should the application close?
Yes
No
Unload resources and return back to caller
Is the GUI rendering?
No
Handle window resize events
Yes
Is the GUI in normal mode?
Yes
Has the user clicked on the image?
Yes
Handle the user clicking to navigate
No
No
Handle the user pressing a keyboard key
Deselect any selected textbox
Is the GUI rendering?
No
GUI mode
Load dialog mode
Save dialog mode
Text dialog mode
Handle 'OK' button press
Handle save dialog buttons
Yes
Normal mode
Update the image buffer from the renderer
Handle all other buttons
Handle load dialog buttons
Is the render finished?
No
Did the GUI handle a button press?
Yes
Yes
Deselect any selected textbox
No
Flag the GUI as no longer rendering
Update the displayed render percentage
Draw the interface
No
Did the rendering parameters change?
Build the preview render
Yes

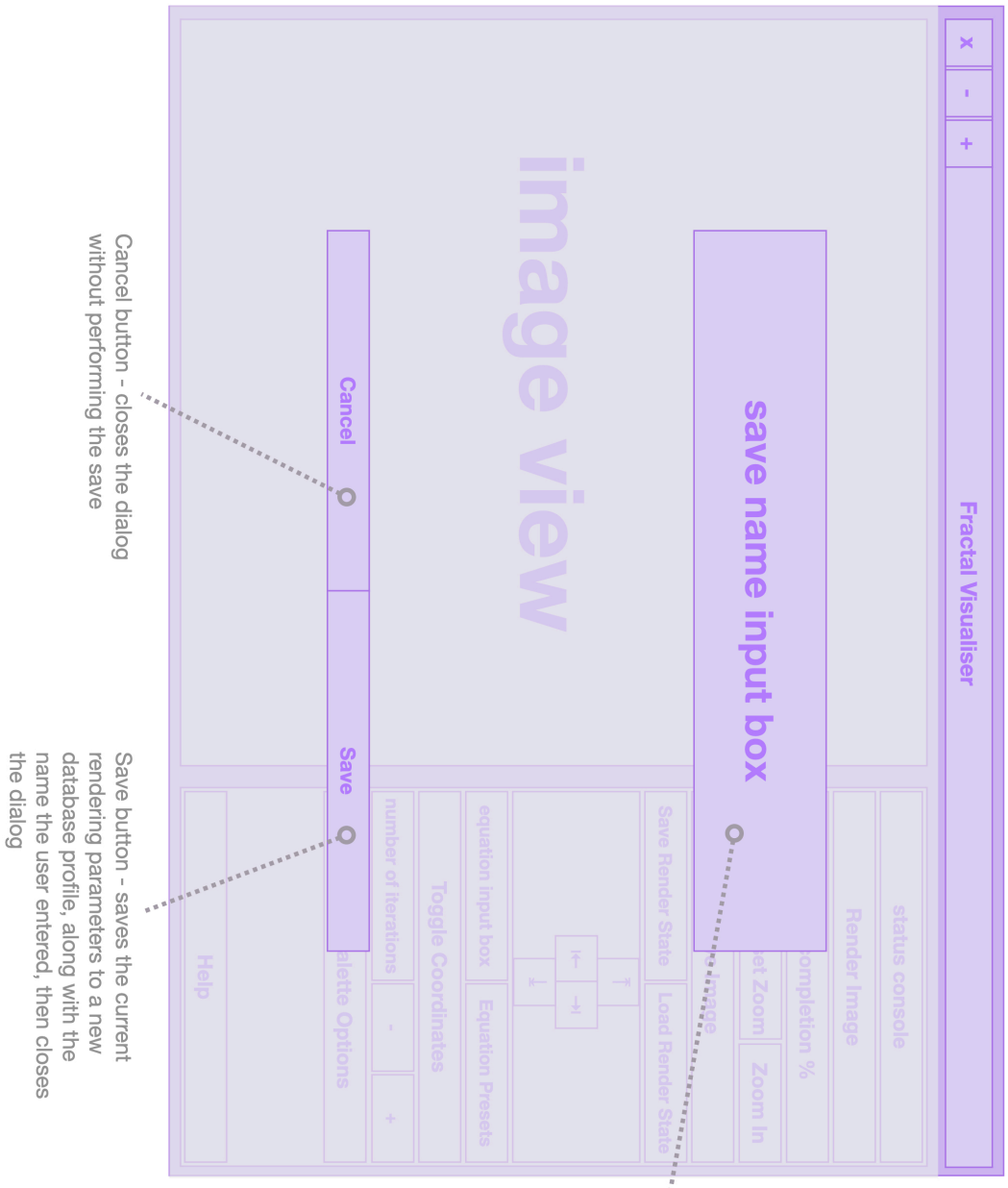

Generate image/build preview render
Is this renderer already rendering?
Yes
Return
No
Is the equation valid?
No
Return
Yes
Flag the renderer as now rendering
Clear the thread pool
Instantiate the specified number of rendering threads
Jump each one to...
Initialise constants (scale factors and offsets)
Get an uncompleted pixel from the image
Initialise Z and C values with coordinate
Calculate coordinates in the complex space
Initialise counter to 1
Evaluate equation substituting in Z and C values
Assign result back to Z variable
Is modulus of Z > 2?
No
Increment counter
Yes
Assign counter value to pixel in image data
Are there more pixels left to compute?
Yes
No
Has the caller requested to wait for the render to finish?
Yes
Wait for all threads to complete and exit
Have all other threads already finished executing?
No
Yes
Flag the renderer as no longer rendering
No
Return

Interface Block-out

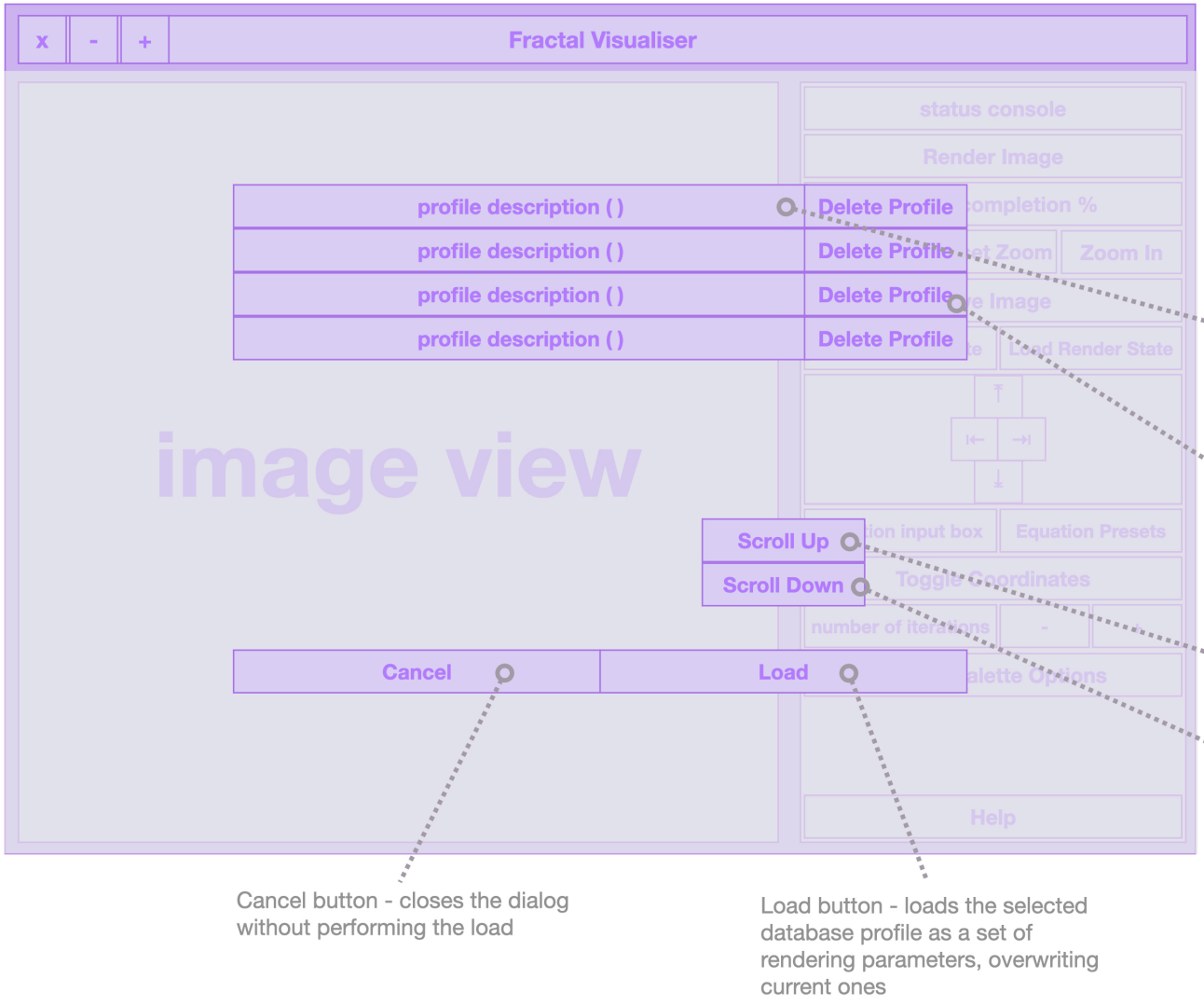
Base GUI State



Database Save Dialog GUI State



Database Load Dialog GUI State



List of profiles showing their descriptions, and which one is selected
Each list item is a button which selects the profile it represents when clicked

Delete Profile button - each profile is accompanied by a button which deletes it from the database when clicked

Scroll Up button - scrolls the profile list view one item further towards the top

Scroll Down button - scrolls the profile list view one item further towards the bottom

Cancel button - closes the dialog without performing the load

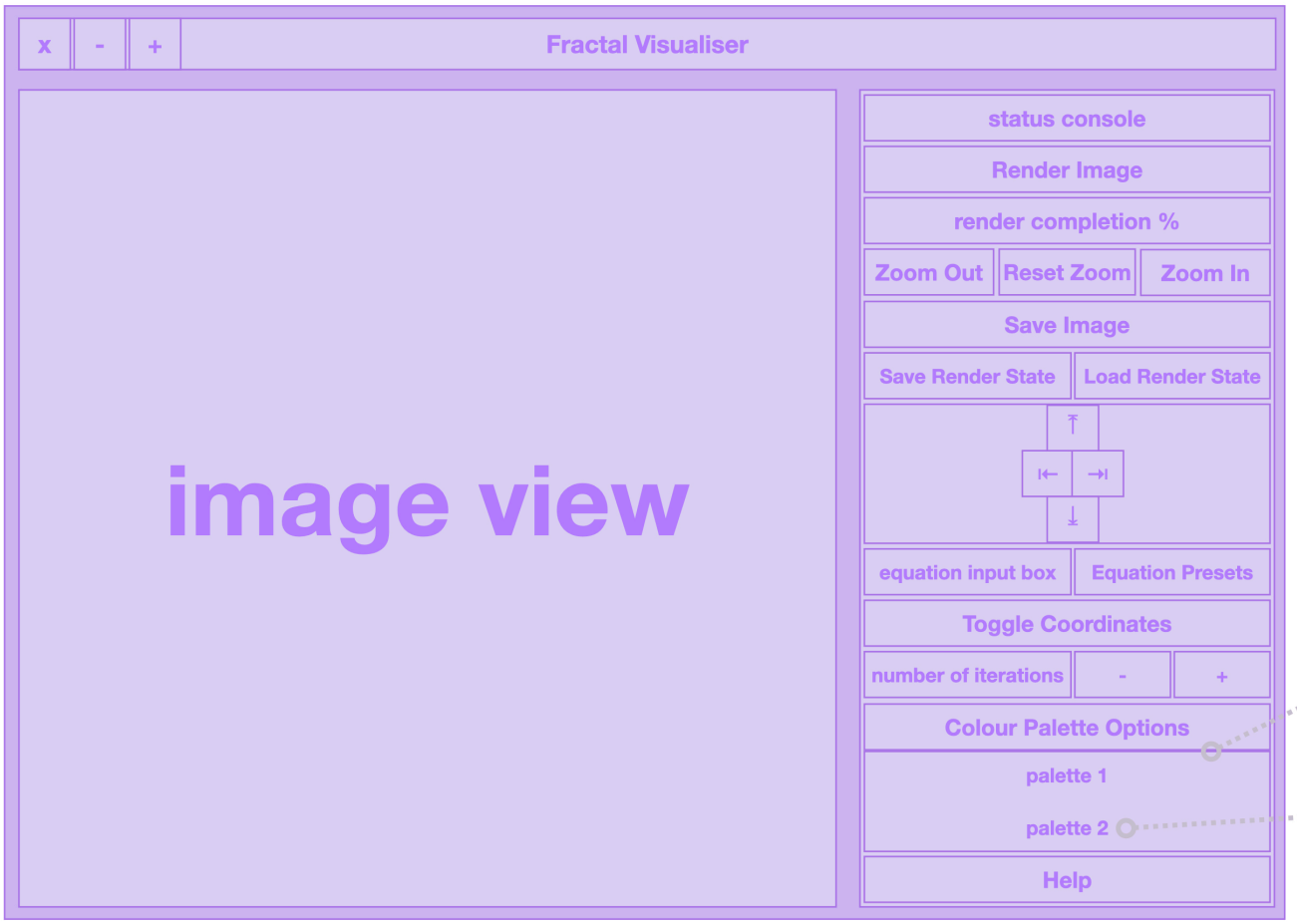
Load button - loads the selected database profile as a set of rendering parameters, overwriting current ones

Equation Preset Dialog GUI State

When the Equation Presets button is clicked, a modal dropdown will appear, containing a button for each equation preset available

Clicking any one of these will close the dialog and make this preset the currently active equation being used by the renderer

Colour Palette Dialog GUI State



When the Colour Palette Options button is clicked, a modal dropdown will appear, containing a button for each colour palette available to choose from

Clicking any one of these will close the dialog and make this palette the currently active palette being used to interpret rendered images

Algorithms & Data Structures

Parsing A Mathematical Expression with Recursion

This is one of the most difficult elements of this project, simply due to the variability of possible cases because this is a complex user-specified parameter.

Equations will be stored using **Reverse Polish Notation**, also known as **postfix notation**. This can be easily stored in an array, and can then be evaluated using a **stack** method.

Extracting the equation from a string is much harder however. I decided to do this in a step-by-step process, as can be seen in the program flow diagram. The string is first cleaned to remove whitespace, then checked to detect errors and invalid equations which would otherwise cause the parser to encounter problems. The parser can then step through the string and convert the sequence of characters into a sequence of intermediate tokens, which are much easier to process afterwards.

This is done by tracking the type of the character currently under inspection (i.e. whether it is a digit, letter, '.', operation, etc) and using changes in the current character type to decide when a new token begins (e.g. when the last character was a digit and the next is a '+', the parser can determine that a numerical constant has finished being read and that an operation token has just begun).

Handling brackets is the ideal situation to use a **recursive function**; this can be done by simply extracting the contents of each set of brackets, and feeding it back into the same tokenising function. The result can then be inserted into the resultant array of tokens of the outer function (and so on as far as necessary until the base case, the outermost function call being performed over the whole equation, is reached again), so that the entire equation is processed including all sets of brackets.

The next step of this is to turn the sequence of 'intermediate' tokens into a finalised array of tokens formatted as postfix notation. Postfix notation is laid out with the operator after the operands for each binary operation.

Before this finalising step, mathematical processing must be performed, which means obeying mathematical rules, specifically the BIDMAS rule for order of operations. This is done simply by iterating over the sequence of tokens and pairing them up over an operation, and wrapping this in brackets (making post-processing of the equation very easy). This is done in multiple passes, one per operation type, so first indices, then division, and so on, following the order of operations in mathematics.

However, another issue must be fixed as well. A common practice in mathematical notation is to use 'ab' instead of writing 'a * b'. This is known as implicit multiplication, and for the purposes of the equation parser it needs to be translated from the shortened form into the explicit form, by replacing instances of the form 'ab' with the form '(a*b)'. Order of operations still needs to be taken into account, so fixing implicit multiplication must be done in between the 'division' and 'multiplication' passes of the BIDMAS simplification phase.

Another mathematical practice which must be observed is the use of '-' as a unary operator. Binary operators are operators (like '+', '-', '*', and so on) which take two arguments and return a single output. Unary operators however only take a single argument. As a result, instances of '-a', where 'a' is any other value (e.g. a bracket, a number, etc), with '0-a'. This transforms the '-' into a binary operator, making it easier to process.

However, once all this is done, the finalising/post-processing of the equation is relatively simple. Because of the simplification phase, we can be sure that the sequence of 'intermediate' tokens will be formed only of a single <token><operation><token>, where each token may be a bracket (which contains another set of <token><operation><token>), a number, or a letter (out of z, x, y, c, a, b, i). In this way, tokens can be easily reordered into postfix notation; if they are simple tokens such as an operation or a number they can simply be copied out, otherwise they will be post-processed **recursively**, once again by calling the post-process function and inserting the result into the postfix token sequence from the outer function (again, this process of recursion is called down each set of brackets, until each sub-call has returned its results and the outermost function has been reached again).

This will generate a simple, linear, no-parsing-required sequence of instructions effectively which can be easily evaluated many times over by the fractal evaluator module.

Evaluating Mathematical Expressions using a Stack

After the majority of the work has been done by the parser, the fractal evaluator has minimal processing to do. This is deliberate, as the module must evaluate the equation many times (a maximum of $ImageSize * ImageSize * IterationLimit$ times, which on an average screen with average settings is very large). This is done by reading along the array of tokens produced by equation parser and treating each as an instruction.

A **stack** will be used to keep track of the computation state. Different tokens are treated differently:

- Number tokens will push a constant onto the stack with the value specified
- Letter tokens will be replaced with the relevant value (e.g. the letter token 'z' will be replaced with the current value for the z variable, see analysis for explanation of fractals) and push-ed onto the stack
- Operation tokens will be executed over the top two items on the stack. These top items will be pop-ed and the result of the operation will be pushed
- At the end of the token sequence, there will be only one value left on the stack, which can be copied off and returned.

This is done many times over for each pixel, until the computed value z is greater than the threshold, which is typically set at 2. Again, see analysis for explanation.

A major optimisation which can be made for this system is for built-in equation presets. The speed of evaluating equations in the way described above is limited by the overhead of performing the stack operations. However, if the equation being used is already known (i.e. it is one of the application's presets) evaluating equations in this way is unnecessary, and so instead for the equation presets (such as Mandelbrot and Julia sets) the evaluation function can be hard-coded and switched to at runtime, providing a major performance boost. Such hard-coded calculations avoid the overheads associated with stack operations and iterating over an array, and can also be optimised considerably in terms of machine code at compile-time by the compiler.

Dividing Up Workload with Multi-threading

Generating the entire fractal image requires a lot of processing, and would take an extremely long time on a single execution thread. Nearly all modern computers have more than one logical processor, meaning they have the capacity to run multiple operations simultaneously, allowing multiple pixels to be computed at the same time. Excluding overheads, this would effectively divide required computation time for the whole image by the number of logical processors, vastly increasing performance (especially on more powerful machines).

Multi-threading is key to achieving as realtime a result as possible. The command line arguments will allow the user to specify the number of worker threads to use, whilst the GUI mode will automatically detect the optimal number of threads to use for the given host system and configure the rendering environments accordingly.

From here, the workload of rendering the image must be divided up between the worker threads. The mechanism to be used is very simple: a custom image management class keeps track of not only the image data containing computed pixel values (i.e. the outputs for each pixel from the evaluator module), but also of whether or not a pixel needs to be calculated or not. This way, each worker thread simply requests a new pixel coordinate from the image manager, which would then mark that pixel as 'in-progress'. The image class should keep track of which pixel should be assigned next to minimise wait times when threads request new pixels to compute. After each pixel is computed, it can be assigned back to the image data of the image class, which will flag the pixel as completed at the same time.

Some locking is required on the image manager class to prevent collisions, but this will be minimised in order to maintain performance without encountering collisions.

When the image is complete, the threads can be automatically halted as computation is complete.

Generating Colours with a Colour Converter

In order to display the fractals, they must be converted from their ‘raw’ form (stored simply as the number of iterations required for each pixel) into a ‘prettified’ colourful image. The solution will provide a set of several different colour presets, which algorithmically convert the raw data into a 32bit RGBA image which can be

displayed. Although not every colour palette will use this algorithm, a **colour conversion** algorithm must be implemented, to convert between different colour formats.

One format for representing colour is ‘HSV’ - Hue, Saturation, Value. Hue represents the particular colour group of a colour, measured around the colour wheel. Saturation represents how strong that colour is (i.e. between pure hue and pure white). Value represents how light the colour is. This format gives good artistic control over colour, as different parameters can be changed more easily (for instance, you can make a colour ‘lighter’ with a single value, without having to manually increase each RGB channel).

Another format is ‘RGB’ - Red, Green, Blue. This format balances the strengths of three components, red, green and blue primary colours to produce another colour (i.e. all colours fully shown is white). This format is more common for representing colour by computers, and is used to render images to the screen, due to the way physical pixels work (they balance the brightness of red, green and blue LEDs).

For images to be displayed, they must be in RGBA format. The A here represents Alpha, which is simply the opacity of a pixel and here is set to always be 100% (fully opaque). However, handling colours as HSV data is much easier to produce nice-looking gradients between colours. Thus, there is a need to write a **mathematical colour conversion algorithm**, to convert HSV defined colours into RGBA colours which can be displayed on the screen. For this the solution will include a simple HSV-to-RGBA function, which implements the standard mathematical algorithm for such an operation.

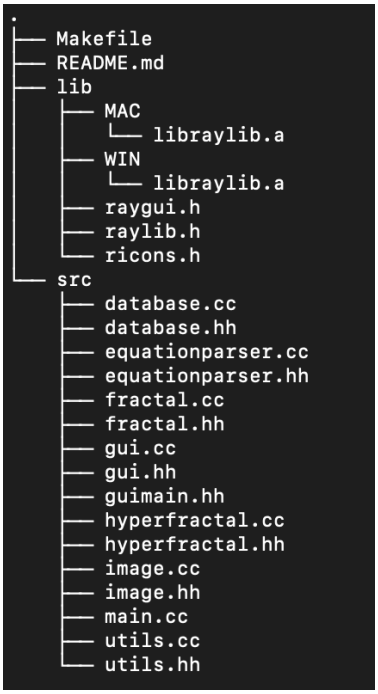
Technical Solution

On the following pages are inserted the source files for the implementation, but not including the Makefile. A configured, buildable version of the project is available on https://github.com/JkyProgrammer/HyperFractal.

Each of the files below has been formatted with a comment at the top which states the name of the file it originates from.

Mini-Contents for Implementation

On the right is the directory structure for the project, including libraries and source files. See the contents table below for the page numbers of specific source files in the following insert.



```cpp
// src/database.cc

#include "database.hh"

#include <vector>
#include <string>
#include <iostream>
#include <fstream>

using namespace std;

/**
 * @brief Modify a string so that it can be written to CSV properly. This means replacing double quotes with a pair of consecutive double quotes
 *
 * @param s Input string
 * @return A ready-to-write result string
 */
std::string HFractalDatabase::forCSVInner (std::string s) {
    string fixed_quotes = "";
    for (char c : s) {
        fixed_quotes += c;
        if (c == '\"') fixed_quotes += '\"';
    }
    fixed_quotes = "\"" + fixed_quotes + "\"";
    return fixed_quotes;
}

/**
 * @brief Break a record from a CSV file into a sequence of raw string fields
 *
 * @param line Line from CSV file to process
 * @return std::vector of raw string fields (require additional processing)
 */
std::vector<std::string> HFractalDatabase::componentify (std::string line) {
    vector<string> ret_value;
    string current_component;
    int start_index = line.find ("\"");
    int end_index = -1;
    while (start_index < line.length()) {
        end_index = line.find ("\",\"", start_index);
```

```
41          if (end_index == string::npos) end_index = line.find ("\"",
start_index+1);
42          current_component = line.substr (start_index+1, end_index-
(start_index+1));
43          current_component = fixDoubleQuote (current_component);
44          ret_value.push_back (current_component);
45          start_index = end_index+2;
46      }
47      return ret_value;
48  }
49
50  /**
51   * @brief Remove double-double quotes from a string. Effectively the reverse
of forCSVInner
52   *
53   * @param s String field to process
54   * @return Cleaned-up field string
55   */
56  std::string HFractalDatabase::fixDoubleQuote (std::string s) {
57      string result = "";
58      char current_char = '\0';
59      char next_char = '\0';
60      for (int i = 0; i < s.length(); i++) {
61          current_char = s[i];
62          next_char = (i+1 < s.length()) ? s[i+1] : '\0';
63          result.push_back (current_char);
64          if (current_char == '\"' && next_char == '\"') i++;
65      }
66      return result;
67  }
68
69  /**
70   * @brief Construct a database instance using a base path to CSV files
71   *
72   * @param path Base path to the target CSV database. Individual tables must
be stored as separate CSV files, so the base path is modified provide paths
to the invidiual CSV files
73   */
74  HFractalDatabase::HFractalDatabase (std::string path) {
75      // Generate and assign the base path
76      int cutoff = path.find(".csv");
77      db_path = path.substr (0, cutoff);
78
```

```
79      // Try to read the database, failing that write a new one, failing that,
    error out
80      if (!read()) if (!commit ()) {
81          throw new std::runtime_error ("unable to create database");
82      }
83  }
84
85  HFractalDatabase::HFractalDatabase () {}
86
87  /**
88   * @brief Get a list of config profile descriptions paired with their IDs.
    Allows the GUI to easily grab a profile summary without having to fetch all
    the data one-by-one
89   *
90   * @return std::vector of pairs of ID and string values
91   */
92  std::vector<std::pair<long, std::string>>
    HFractalDatabase::getConfigDescriptions () {
93      std::vector<std::pair<long, std::string>> ret_val;
94      for (auto conf : configs) {
95          std::pair<long, std::string> desc_pair;
96          desc_pair.first = conf.first;
97          desc_pair.second = (
98              conf.second->name
99              + " ("
100             + conf.second->equation
101             + ")");
102         ret_val.push_back (desc_pair);
103     }
104     return ret_val;
105 }
106
107 /**
108  * @brief Function to get a configuration profile by its ID
109  *
110  * @param id The ID of the profile
111  * @return A pointer to the configuration profile
112  */
113 HFractalConfigProfile* HFractalDatabase::getConfig (long id) {
114     HFractalConfigProfile *ret = NULL;
115     if (configs.count(id) != 0) ret = configs[id];
116     return ret;
117 }
```

```
118
119 /**
120  * @brief Function to get a user profile by its ID
121  *
122  * @param id The ID of the profile
123  * @return A pointer to the user profile
124  */
125 HFractalUserProfile* HFractalDatabase::getUser (long id) {
126     HFractalUserProfile *ret = NULL;
127     if (users.count(id) != 0) ret = users[id];
128     return ret;
129 }
130
131 /**
132  * @brief Insert a configuration profile into the database. Automatically
    generates and assigns a unique ID
133  *
134  * @param c Pointer to the config profile being inserted
135  * @return Generated ID of the profile
136  */
137 long HFractalDatabase::insertConfig (HFractalConfigProfile* c) {
138     long max_id = -1;
139     for (pair<long, HFractalConfigProfile*> p : configs)
140         if (p.first > max_id) max_id = p.first;
141
142     c->profile_id = max_id+1;
143     configs.emplace (c->profile_id, c);
144     return c->profile_id;
145 }
146
147 /**
148  * @brief Insert a user profile into the database. Automatically generates
    and assigns a unique ID
149  *
150  * @param u Pointer to the user profile being inserted
151  * @return Generated ID of the profile
152  */
153 long HFractalDatabase::insertUser (HFractalUserProfile* u) {
154     long max_id = -1;
155     for (pair<long, HFractalUserProfile*> p : users)
156         if (p.first > max_id) max_id = p.first;
157
```

```
    u->user_id = max_id+1;
    users.emplace (u->user_id, u);
    return u->user_id;
}

/**
 * @brief Delete a config profile record from the database
 *
 * @param id ID of the profile to be deleted
 * @return True if the delete succeeded, False if the record did not exist
 */
bool HFractalDatabase::removeConfig (long id) {
    return configs.erase (id);
}

/**
 * @brief Delete a user profile record from the database
 *
 * @param id ID of the profile to be deleted
 * @return True if the delete succeeded, False if the record did not exist
 */
bool HFractalDatabase::removeUser (long id) {
    return users.erase (id);
}

/**
 * @brief Write out the contents of the cached database to CSV files.
 *
 * @return True if the write succeeded, False if it failed
 */
bool HFractalDatabase::commit () {
    // Create path strings and file streams
    string db_path_configs = db_path + "_configs.csv";
    string db_path_users = db_path + "_users.csv";
    ofstream db_file_configs (db_path_configs.c_str());
    ofstream db_file_users (db_path_users.c_str());

    // If the file streams are open, write data
    if (db_file_configs.is_open() && db_file_users.is_open()) {
        // Iterate over config files
        for (auto copair : configs) {
            HFractalConfigProfile* config = copair.second;
```

```
            string line = "";
            line += forCSV (config->profile_id) + ",";
            line += forCSV (config->x_offset) + ",";
            line += forCSV (config->y_offset) + ",";
            line += forCSV (config->zoom) + ",";
            line += forCSV (config->iterations) + ",";
            line += forCSV (config->equation) + ",";
            line += forCSV (config->name) + ",";
            line += forCSV (config->palette) + ",";
            line += forCSV (config->user_id);
            db_file_configs << line.c_str() << endl;
        }

        // Iterate over user files
        for (auto upair : users) {
            HFractalUserProfile* user = upair.second;
            string line = "";
            line += forCSV (user->user_id) + ",";
            line += forCSV (user->user_name);
            db_file_users << line.c_str() << endl;
        }

        // Close the files and return success
        db_file_configs.close();
        db_file_users.close();
        return true;
    } else return false; // Return failure
}

/**
 * @brief Read the contents of CSV files into the cached database
 *
 * @return True if the read succeeded, False if it failed
 */
bool HFractalDatabase::read () {
    // Create paths and file streams
    string db_path_configs = db_path + "_configs.csv";
    string db_path_users = db_path + "_users.csv";
    ifstream db_file_configs (db_path_configs.c_str());
    ifstream db_file_users (db_path_users.c_str());

    // If the file streams are open, read data
```

```
242     if (db_file_configs.is_open() && db_file_users.is_open()) {
243         string line;
244         int line_number = 0;
245 
246         configs.clear();
247         // Read config profiles
248         HFractalConfigProfile* config;
249         while (getline (db_file_configs, line)) {
250             try {
251                 // Attempt to convert the record to a config profile
252                 vector<string> components = componentify (line);
253                 config = new HFractalConfigProfile ();
254                 config->profile_id = stol(components[0]);
255                 config->x_offset = stold(components[1]);
256                 config->y_offset = stold(components[2]);
257                 config->zoom = stold(components[3]);
258                 config->iterations = stoi(components[4]);
259                 config->equation = components[5];
260                 config->name = components[6];
261                 config->palette = stoi(components[7]);
262                 config->user_id = stol(components[8]);
263                 configs.emplace (config->profile_id, config);
264             } catch (std::invalid_argument e) {
265                 // Print a console error if a line could not be read
266                 cout << "Failed to read config profile on line " <<
line_number << endl;
267             }
268             line_number++;
269         }
270 
271         line_number = 0;
272 
273         // Read user profiles
274         users.clear();
275         HFractalUserProfile* user;
276         while (getline (db_file_users, line)) {
277             try {
278                 // Attempt to convert the record to a user profile
279                 vector<string> components = componentify (line);
280                 user = new HFractalUserProfile ();
281                 user->user_id = stol(components[0]);
282                 user->user_name = components[1];
```

```
283                 users.emplace (user->user_id, user);
284             } catch (std::invalid_argument e) {
285                 // Print a console error if a line could not be read
286                 cout << "Failed to read user profile on line " <<
line_number << endl;
287             }
288             line_number++;
289         }
290         // Return success
291         return true;
292     } else return false; // Return failure
293 }
294 
295 /**
296  * @brief Generate a CSV-happy string from a given input
297  *
298  * @param s String input
299  * @return CSV-writeable string
300  */
301 std::string HFractalDatabase::forCSV (std::string s) {
302     return forCSVInner (s);
303 }
304 
305 /**
306  * @brief Generate a CSV-happy string from a given input
307  *
308  * @param l Long integer input
309  * @return CSV-writeable string
310  */
311 std::string HFractalDatabase::forCSV (long l) {
312     return forCSVInner (to_string(l));
313 }
314 
315 /**
316  * @brief Generate a CSV-happy string from a given input
317  *
318  * @param ld Long double input
319  * @return CSV-writeable string
320  */
321 std::string HFractalDatabase::forCSV (long double ld) {
322     return forCSVInner (to_string(ld));
323 }
```

```
324
325 /**
326  * @brief Generate a CSV-happy string from a given input
327  *
328  * @param i Integer input
329  * @return CSV-writeable string
330  */
331 std::string HFractalDatabase::forCSV (int i) {
332     return forCSVInner (to_string(i));
333 }
```

```
1  // src/database.hh
2
3  #ifndef DATABASE_H
4  #define DATABASE_H
5
6  #include <string>
7  #include <vector>
8  #include <unordered_map>
9  #include <cstring>
10
11 // Struct describing the Config Profile record type
12 struct HFractalConfigProfile {
13     long profile_id; // Primary key
14
15     long double x_offset;
16     long double y_offset;
17     long double zoom;
18     int iterations;
19     std::string equation;
20     std::string name;
21     int palette;
22     long user_id; // Foreign key of HFractalUserProfile
23
24     HFractalConfigProfile () { memset (this, 0,
   sizeof(HFractalConfigProfile)); }
25 };
26
27 // Struct describing the User Profile record type
28 struct HFractalUserProfile {
29     long user_id; // Primary key
30
31     std::string user_name;
32
33     HFractalUserProfile () { memset (this, 0, sizeof(HFractalUserProfile)); }
34 };
35
36 // Class for managing the database of users and configurations and providing
   access to data for the GUI
37 class HFractalDatabase {
38 private:
39     std::string db_path; // Base path to the database
40     std::unordered_map<long, HFractalConfigProfile*> configs; // Map of
```

```
    config profiles against their IDs
41      std::unordered_map<long, HFractalUserProfile*> users; // Map of user
    profiles against their IDs
42      static std::string forCSVInner (std::string); // Static function to
    convert a string into a form CSVs will read/write properly
43      std::vector<std::string> componentify (std::string); // Break a line of
    CSV into fields
44      std::string fixDoubleQuote (std::string); // Remove double quotes in
    strings read from CSV file
45
46      static std::string forCSV (std::string); // Generate a string which can
    be written to a CSV file as a field of a record
47      static std::string forCSV (long); // Generate a string which can be
    written to a CSV file as a field of a record
48      static std::string forCSV (int); // Generate a string which can be
    written to a CSV file as a field of a record
49      static std::string forCSV (long double); // Generate a string which can
    be written to a CSV file as a field of a record
50  public:
51      HFractalDatabase (std::string); // Initialise the database from a given
    base path
52      HFractalDatabase (); // Dead initialiser for implicit instantiation
53
54      std::vector<std::pair<long, std::string>> getConfigDescriptions (); //
    Generate a list of ID and description pairs for the GUI to display
55      HFractalConfigProfile* getConfig (long); // Get a pointer to the config
    profile with a given ID
56      HFractalUserProfile* getUser (long); // Get a pointer to the user profile
    with a given ID
57
58      long insertConfig (HFractalConfigProfile*); // Insert a new config record
    and return its ID
59      long insertUser (HFractalUserProfile*); // Insert a new user record and
    return its ID
60
61      bool removeConfig (long); // Remove a config record by ID
62      bool removeUser (long); // Remove a user record by ID
63
64      bool commit (); // Write the contents of the cached database out to CSV
    files
65      bool read (); // Read the contents of a CSV file into the database cache
66  };
67
68  #endif
```

```cpp
// src/equationparser.cc

#include "equationparser.hh"

#include <vector>

using namespace std;

/**
 * @brief Clean whitespace out of the input string
 *
 * @param s Input string
 * @return Cleaned string
 */
string HFractalEquationParser::epClean (string s) {
    string ret_val = "";
    for (char c : s) if (c != ' ') ret_val += c;
    return ret_val;
}

/**
 * @brief Check that the input string is valid for the HFractalEquation parser to analyse. Checks for the following and returns an enum value accordingly:
 *
 * 0 - No error found
 * 1 - Bracket error: '()', '(' not equal number to ')', ')...('
 * 2 - Operation error: '**', '*-' or any other repetition of an operation
 * 3 - Implicit multiplication error: 'z2' rather than correct synax '2z'
 * 4 - Floating point error: '.46', '34.'
 * 5 - Unsupported character error: '$', 'd', or any other character not accounted for
 *
 * @param s Input string
 * @return Either the reference of the first error detected or SUCCESS if no error is found
 */
EP_CHECK_STATUS HFractalEquationParser::epCheck (string s) {
    int bracket_depth = 0;
    char c_last = '\0';
    int index = 0;
    for (char c : s) {
        switch (c) {
```

```
40          case '(':
41              bracket_depth++;
42              if (c_last == '.') return FPOINT_ERROR;
43              break;
44          case ')':
45              bracket_depth--;
46              if (c_last == '(') return BRACKET_ERROR;
47              if (c_last == '.') return FPOINT_ERROR;
48              break;
49          case 'z':
50          case 'c':
51          case 'a':
52          case 'b':
53          case 'x':
54          case 'y':
55          case 'i':
56              if (c_last == '.') return FPOINT_ERROR;
57              break;
58          case '*':
59          case '/':
60          case '+':
61          case '^':
62              if (c_last == '*' || c_last == '/' || c_last == '-' || c_last ==
'+' || c_last == '^') return OPERATION_ERROR;
63              if (c_last == '.') return FPOINT_ERROR;
64              if (index == 0 || index == s.length()-1) return OPERATION_ERROR;
65              break;
66          case '-':
67              if (c_last == '-') return OPERATION_ERROR;
68              if (c_last == '.') return FPOINT_ERROR;
69              if (index == s.length()-1) return OPERATION_ERROR;
70              break;
71          case '0':
72          case '1':
73          case '2':
74          case '3':
75          case '4':
76          case '5':
77          case '6':
78          case '7':
79          case '8':
80          case '9':
```

```
81              if (c_last == 'z' || c_last == 'c' || c_last == 'i' || c_last == 'a' || c_last == 'b' || c_last == 'x' || c_last == 'y') return IMULT_ERROR;
82              break;
83          case '.':
84              if (!(c_last == '0' || c_last == '1' || c_last == '2' || c_last == '3' || c_last == '4' || c_last == '5' || c_last == '6' || c_last == '7' || c_last == '8' || c_last == '9')) return FPOINT_ERROR;
85              break;
86          default:
87              return UNSUPCHAR_ERROR;
88              break;
89          }
90
91          c_last = c;
92          index++;
93          if (bracket_depth < 0) return BRACKET_ERROR;
94      }
95
96      if (bracket_depth != 0) return BRACKET_ERROR;
97      return SUCCESS;
98  }
99
100 /**
101  * @brief Break string into tokens for processing. Assumes epCheck has been called on `s` previously and that this has returned 0
102  *
103  * @param s Input string
104  * @return std::vector of tokens
105  */
106 vector<IntermediateToken> HFractalEquationParser::epTokenise (string s) {
107     vector<IntermediateToken> token_vec;
108     string current_token = "";
109     int current_token_type = -1;
110     bool is_last_run = false;
111     bool is_singular_token = false; // Informs the program whether the token is a single-char token
112
113     for (int i = 0; i < s.length(); i++) {
114         char current_char = s[i];
115         int char_token_type = -1;
116
117         // Decide the type of the current character
118         switch (current_char) {
```

```
        case '0':
        case '1':
        case '2':
        case '3':
        case '4':
        case '5':
        case '6':
        case '7':
        case '8':
        case '9':
        case '.':
            char_token_type = 0;
            break;
        case 'z':
            char_token_type = 1;
            break;
        case 'c':
            char_token_type = 2;
            break;
        case 'a':
            char_token_type = 6;
            break;
        case 'b':
            char_token_type = 7;
            break;
        case 'x':
            char_token_type = 8;
            break;
        case 'y':
            char_token_type = 9;
            break;
        case 'i':
            char_token_type = 3;
            break;
        case '*':
        case '/':
        case '-':
        case '+':
        case '^':
            char_token_type = 4;
            break;
        case '(':
```

```
161             char_token_type = 5;
162             break;
163         default:
164             break;
165         }
166 
167         // Save the current token the token vector
168         if (char_token_type != current_token_type || is_last_run ||
is_singular_token) {
169             is_singular_token = false;
170             if (current_token.length () > 0) {
171                 IntermediateToken token;
172                 switch (current_token_type) {
173                 case 0:
174                     token.type = INT_NUMBER;
175                     token.num_val = stod (current_token);
176                     break;
177                 case 1:
178                 case 2:
179                 case 3:
180                 case 6:
181                 case 7:
182                 case 8:
183                 case 9:
184                     token.type = INT_LETTER;
185                     token.let_val = current_token[0];
186                     break;
187                 case 4:
188                     token.type = INT_OPERATION;
189                     token.op_val = current_token[0];
190                     break;
191                 case 5:
192                     token.type = INT_BRACKET;
193                     token.bracket_val = epTokenise (current_token);
194                 default:
195                     break;
196                 }
197                 token_vec.push_back (token);
198             }
199             current_token = "";
200             current_token_type = char_token_type;
201 
```

```
            if (is_last_run) break;
        }

        // Jump automatically to the end of the brackets, recursively processing their contents
        if (char_token_type == 5) {
            int bracket_depth = 1;
            int end = -1;
            for (int j = i+1; j < s.length(); j++) {
                if (s[j] == '(') bracket_depth++;
                if (s[j] == ')') bracket_depth--;
                if (bracket_depth == 0) { end = j; break; }
            }

            current_token = s.substr (i+1, end-(i+1));
            i = end;
        } else { // Otherwise append the current character to the current token
            current_token += s[i];
        }

        // Mark a, b, c, x, y, z, and i as singular
        if ((char_token_type >= 1 && char_token_type <= 3) || (char_token_type >= 6 && char_token_type <= 9)) {
            is_singular_token = true;
        }

        // If we've reached the end of the string, jump back and mark it as a last pass in order to save the current token
        if (i == s.length()-1) {
            is_last_run = true;
            i--;
        }
    }

    // Check through and repair any `-...` expressions to be `0-...`
    for (int i = 0; i < token_vec.size(); i++) {
        if (token_vec[i].type == INT_OPERATION && token_vec[i].op_val == '-') {
            if (i == 0 || (i > 0 && token_vec[i-1].type == INT_OPERATION)) {
                IntermediateToken bracket;
                bracket.type = INT_BRACKET;
                int bracket_length = 1;
```

```
                bracket.bracket_val.push_back ({
                    .type = INT_NUMBER,
                    .num_val = 0
                });
                bracket.bracket_val.push_back ({
                    .type = INT_OPERATION,
                    .op_val = '-'
                });
                while (i+bracket_length < token_vec.size() &&
token_vec[i+bracket_length].type != INT_OPERATION) {
                    bracket.bracket_val.push_back
(token_vec[i+bracket_length]);
                    bracket_length++;
                }

                for (int tmp = 0; tmp < bracket_length; tmp++)
token_vec.erase (next(token_vec.begin(), i));
                token_vec.insert (next(token_vec.begin(), i), bracket);
                i -= bracket_length-1;
            }
        }
    }

    return token_vec;
}

/**
 * @brief Search for and replace implicit multiplication (adjacent non-
operation tokens such as '5z' or '(...)(...)') with explicit multiplication
 *
 * @param token_vec Token vector to fix
 * @return Token vector with no implicit multiplication
 */
vector<IntermediateToken> HFractalEquationParser::epFixImplicitMul
(vector<IntermediateToken> token_vec) {
    vector<IntermediateToken> result = token_vec;
    for (int i = 0; i < result.size()-1; i++) {
        IntermediateToken t1 = result[i];
        IntermediateToken t2 = result[i+1];

        // Fix implicit multiplication within brackets
        if (t1.type == INT_BRACKET) {
            result[i].bracket_val = epFixImplicitMul (t1.bracket_val);
```

```
278             t1 = result[i];
279         }
280 
281         // Detect two adjacent tokens, where neither is an operation
282         if (!(t1.type == INT_OPERATION || t2.type == INT_OPERATION)) {
283             result.erase (next(result.begin(), i));
284             result.erase (next(result.begin(), i));
285             IntermediateToken explicit_mul;
286             explicit_mul.type = INT_BRACKET;
287             explicit_mul.bracket_val.push_back (t1);
288             explicit_mul.bracket_val.push_back ({
289                 .type = INT_OPERATION,
290                 .op_val = '*'
291             });
292             explicit_mul.bracket_val.push_back (t2);
293 
294             result.insert (next(result.begin(), i), explicit_mul);
295             i--;
296         }
297     }
298 
299     IntermediateToken last = result[result.size()-1];
300     if (last.type == INT_BRACKET) {
301         last.bracket_val = epFixImplicitMul (last.bracket_val);
302         result[result.size()-1] = last;
303     }
304 
305     return result;
306 }
307 
308 /**
309  * @brief Ensure BIDMAS (Brackets Indices Division Multiplication Addition
    Subtraction) order mathematical evaluation by search-and-replacing each with
    brackets
310  *
311  * @param token_vec Token vector to simplify
312  * @return Token vector which requires only sequential evaluation
313  */
314 vector<IntermediateToken> HFractalEquationParser::epSimplifyBidmas
    (vector<IntermediateToken> token_vec, bool first_half) {
315     vector<IntermediateToken> result = token_vec;
316     // Recurse down brackets
317     for (int i = 0; i < result.size(); i++) {
```

```
318         if (result[i].type == INT_BRACKET) {
319             result[i].bracket_val = epSimplifyBidmas (result[i].bracket_val,
first_half);
320         }
321     }
322 
323     // Order of operations:
324     //           IDMAS
325     string ops = "^/*+-";
326 
327     // First half allows the parser to make indices and division explicit,
then process implicit multiplication, and then to process other operations
328     // This allows us to maintain BIDMAS even with explicit multiplication
(e.g. `5z^2` should be `5*(z^2)` and not `(5*z)^2`)
329     if (first_half) {
330         ops = "^/";
331     } else {
332         ops = "*+-";
333     }
334 
335     if (result.size() < 5) return result;
336 
337     // Search and replace each sequentially
338     for (char c : ops) {
339         for (int t_ind = 0; t_ind < result.size()-2; t_ind++) {
340             if (t_ind >= result.size()-2) {
341                 break;
342             }
343             if (result[t_ind+1].type == INT_OPERATION &&
result[t_ind+1].op_val == c) {
344                 IntermediateToken bracket;
345                 bracket.type = INT_BRACKET;
346                 bracket.bracket_val.push_back (result[t_ind]);
347                 bracket.bracket_val.push_back (result[t_ind+1]);
348                 bracket.bracket_val.push_back (result[t_ind+2]);
349 
350                 for (int tmp = 0; tmp < 3; tmp++) result.erase
(next(result.begin(), t_ind));
351                 result.insert (next(result.begin(), t_ind), bracket);
352                 t_ind -= 2;
353             }
354         }
355     }
```

```
356
357     return result;
358 }
359
360 /**
361  * @brief Convert intermediate token vector into usable Reverse Polish
Notation token queue
362  * Ensure that everything else is done before calling this function:
363  *  epClean
364  *  epTokenise
365  *  epSimplifyBidmas first_half=true
366  *  epFixImplicitMul
367  *  epSimplifyBidmas first_half=false
368  * These functions ensure the token vector is ready to be linearly parsed
into Reverse Polish
369  *
370  * @param intermediate Token vector to convert
371  * @return Vector of proper tokens, ready to use in the expression evaluator
372  */
373 vector<Token> HFractalEquationParser::epReversePolishConvert
(vector<IntermediateToken> intermediate) {
374     vector<Token> output;
375
376     IntermediateToken operation = {.op_val = '\0'};
377
378     for (int index = 0; index < intermediate.size(); index++) {
379         IntermediateToken current_intermediate_token = intermediate[index];
380         if (current_intermediate_token.type == INT_OPERATION) {
381             operation.op_val = current_intermediate_token.op_val;
382         } else {
383             // Append to token(s) rp notation
384             if (current_intermediate_token.type == INT_BRACKET) {
385                 vector<Token> inner_result = epReversePolishConvert
(current_intermediate_token.bracket_val);
386                 output.insert (output.end(), inner_result.begin(),
inner_result.end());
387             } else {
388                 output.push_back ({
389                     .type = (TOKEN_TYPE)current_intermediate_token.type,
390                     .num_val = current_intermediate_token.type == INT_NUMBER
? current_intermediate_token.num_val : 0,
391                     .other_val = current_intermediate_token.type ==
INT_LETTER ? current_intermediate_token.let_val : '\0'
392                 });
```

```
393             }
394 
395             // If relevant, append operation to rp notation
396             if (operation.op_val != '\0') {
397                 output.push_back ({
398                     .type = OPERATION,
399                     .other_val = operation.op_val
400                 });
401                 operation.op_val = '\0';
402             }
403         }
404     }
405 
406     return output;
407 }
408 
409 /**
410  * @brief Convert a string mathematical expression into an HFractalEquation class instance using Reverse Polish Notation
411  *
412  * @param sequ String containing a mathematical expression to parse
413  * @return Pointer to an HFractalEquation instance representing the input string
414  */
415 HFractalEquation* HFractalEquationParser::extractEquation (string sequ) {
416     if (sequ.length() < 1) return NULL;
417     string cleaned = epClean (sequ);
418     EP_CHECK_STATUS check_result = epCheck (cleaned);
419     if (check_result != SUCCESS) {
420         return NULL;
421     }
422 
423     vector<IntermediateToken> expression = epTokenise (cleaned);
424 
425     expression = epSimplifyBidmas (expression, true);
426     expression = epFixImplicitMul (expression);
427     expression = epSimplifyBidmas (expression, false);
428 
429     vector<Token> reverse_polish_expression = epReversePolishConvert (expression);
430 
431     return new HFractalEquation (reverse_polish_expression);
432 }
```

```cpp
// src/equationparser.hh

#ifndef EQUATIONPARSER_H
#define EQUATIONPARSER_H

#include <string>
#include <vector>

#include "fractal.hh"

// Enum describing the token type for the intermediate parser
enum INTERMEDIATE_TOKEN_TYPE {
    INT_NUMBER,
    INT_LETTER,
    INT_OPERATION,
    INT_BRACKET
};

// Struct describing the token for the intermediate parser
struct IntermediateToken {
    INTERMEDIATE_TOKEN_TYPE type;
    double num_val;
    char let_val;
    char op_val;
    std::vector<IntermediateToken> bracket_val;
};

// Enum describing the error types from the equation processor checking function
enum EP_CHECK_STATUS {
    SUCCESS,
    BRACKET_ERROR,
    OPERATION_ERROR,
    IMULT_ERROR,
    FPOINT_ERROR,
    UNSUPCHAR_ERROR
};

// Class containing static methods used to parse a string into a postfix token vector
class HFractalEquationParser {
private:
```

```
41      static std::string epClean (std::string); // Preprocess the string to
    remove whitespace
42      static EP_CHECK_STATUS epCheck (std::string); // Check for formatting
    errors in the equation (such as mismatched brackets)
43      static std::vector<IntermediateToken> epTokenise (std::string); // Split
    the string into intermediate tokens
44      static std::vector<IntermediateToken> epFixImplicitMul
    (std::vector<IntermediateToken>); // Remove implicit multiplication
45      static std::vector<IntermediateToken> epSimplifyBidmas
    (std::vector<IntermediateToken>, bool); // Convert BIDMAS rules into explicit
    writing
46      static std::vector<Token> epReversePolishConvert
    (std::vector<IntermediateToken>); // Convert intermediate tokens into a final
    output postfix notation
47
48  public:
49      static HFractalEquation* extractEquation (std::string); // Extract an
    equation containing postfix tokens from a string input
50  };
51
52
53  #endif
```

```cpp
// src/fractal.cc

#include "fractal.hh"

#include <stack>
#include <complex>

#include "utils.hh"

using namespace std;

/**
 * @brief Check if a complex number has tended to infinity. Allows methods which use this check to be implementation independent
 * Tending to infinity is typically defined as |z| > 2, which here is expanded to maximise optimsation
 *
 * @param comp Complex number to check
 * @return True if the number has tended to infinity, False otherwise
 */
bool HFractalEquation::isInfinity (complex<long double> comp) {
    return (comp.real()*comp.real()) + (comp.imag()*comp.imag()) > (long double)4;
}

/**
 * @brief Set the equation preset value
 *
 * @param i Integer representing the preset ID, linked with EQ_PRESETS, or -1 to disable preset mode in this instance
 */
void HFractalEquation::setPreset (int i) {
    is_preset = (i != -1);
    preset = i;
}

/**
 * @brief Parse the Reverse Polish notation Token vector and evaluate the mathematical expression it represents
 *
 * @param z Current value of the z variable to feed in
 * @param c Current value of the c variable to feed in
 * @return Complex number with the value of the evaluated equation
```

```
 */
complex<long double> HFractalEquation::compute (complex<long double> z,
complex<long double> c) {
    stack<complex<long double>> value_stack;

    for (Token t : reverse_polish_vector) {
        if (t.type == NUMBER) {
            // Push number arguments onto the stack
            value_stack.push (t.num_val);
        } else if (t.type == LETTER) {
            // Select based on letter and swap in the letter's value, before
pushing it onto the stack
            switch (t.other_val) {
            case 'z':
                value_stack.push (z);
                break;
            case 'c':
                value_stack.push (c);
                break;
            case 'a':
                value_stack.push (c.real());
                break;
            case 'b':
                value_stack.push (c.imag());
                break;
            case 'x':
                value_stack.push (z.real());
                break;
            case 'y':
                value_stack.push (z.imag());
                break;
            case 'i':
                value_stack.push (complex<long double> (0,1));
                break;
            default:
                break;
            }
        } else if (t.type == OPERATION) {
            // Perform an actual computation based on an operation token
            complex<long double> v2 = value_stack.top(); value_stack.pop();
            complex<long double> v1 = value_stack.top(); value_stack.pop();
            switch (t.other_val) {
```

```
79              case '^':
80                  value_stack.push (pow (v1, v2));
81                  break;
82              case '/':
83                  value_stack.push (v1/v2);
84                  break;
85              case '*':
86                  value_stack.push (v1*v2);
87                  break;
88              case '+':
89                  value_stack.push (v1+v2);
90                  break;
91              case '-':
92                  value_stack.push (v1-v2);
93                  break;
94              default:
95                  break;
96              }
97          }
98      }
99
100     // Return the final value
101     return value_stack.top();
102 }
103
104 /**
105  * @brief Evaluate a complex coordinate (i.e. a pixel) to find the point at which it tends to infinity, by iteratively applying the equation as a mathematical function
106  *
107  * @param c Coordinate in the complex plane to initialise with
108  * @param limit Limit for the number of iterations to compute before giving up, if the number does not tend to infinity
109  * @return Integer representing the number of iterations performed before the number tended to infinity, or the limit if this was reached first
110  */
111 int HFractalEquation::evaluate (complex<long double> c, int limit) {
112     complex<long double> last = c;
113     if (is_preset && preset == EQ_BURNINGSHIP_MODIFIED) {
114         last = complex<long double> (0, 0);
115     }
116
117     int depth = 0;
```

```
118     while (depth < limit) {
119         // Switch between custom parsing mode and preset mode for more
    efficient computing of presets
120         if (!is_preset) {
121             last = compute (last, c); // Slow custom compute
122         } else {
123             // Much faster hard coded computation
124             switch (preset) {
125             case EQ_MANDELBROT:
126                 last = (last*last)+c;
127                 break;
128             case EQ_JULIA_1:
129                 last = (last*last)+complex<long double>(0.285, 0.01);
130                 break;
131             case EQ_JULIA_2:
132                 last = (last*last)-complex<long double>(0.70176, 0.3842);
133                 break;
134             case EQ_RECIPROCAL:
135                 last = complex<long double>(1,0)/((last*last)+c);
136                 break;
137             case EQ_ZPOWER:
138                 last = pow(last,last)+c-complex<long double>(0.5, 0);
139                 break;
140             case EQ_BARS:
141                 last = pow(last, c*c);
142                 break;
143             case EQ_BURNINGSHIP_MODIFIED:
144                 last = pow ((complex<long double>(abs(last.real()),0) -
    complex<long double>(0, abs(last.imag()))),2)+c;
145                 break;
146             default:
147                 break;
148             }
149         }
150         depth++;
151         // Check if the value has tended to infinity, and escape the loop if
    so
152         bool b = isInfinity (last);
153         if (b) break;
154     }
155     return depth;
156 }
157
```

```
158 /**
159  * @brief Initialise with the token sequence in postfix form which this
    class should use
160  *
161  * @param rp_vec Reverse Polish formatted vector of tokens
162  */
163 HFractalEquation::HFractalEquation (vector<Token> rp_vec) {
164     reverse_polish_vector = rp_vec;
165 }
166
167 /**
168  * @brief Base initialiser. Should only be used to construct presets, as the
    equation token vector cannot be assigned after initialisation
169  *
170  */
171 HFractalEquation::HFractalEquation () {}
```

```cpp
// src/fractal.hh

#ifndef FRACTAL_H
#define FRACTAL_H

#include <complex>
#include <vector>

// Enum describing the token type
enum TOKEN_TYPE {
    NUMBER,
    LETTER,
    OPERATION
};

// Struct describing the token
struct Token {
    TOKEN_TYPE type;
    double num_val;
    char other_val;
};

// Class holding the equation and providing functions to evaluate it
class HFractalEquation {
private:
    static bool isInfinity (std::complex<long double> comp); // Check if a complex number has exceeded the 'infinity' threshold
    std::vector<Token> reverse_polish_vector; // Sequence of equation tokens in postfix form

    bool is_preset = false; // Records whether this equation is using an equation preset
    int preset = -1; // Records the equation preset being used, if none, set to -1

public:
    void setPreset (int); // Set this equation to be a preset, identified numerically

    std::complex<long double> compute (std::complex<long double>, std::complex<long double>); // Perform a single calculation using the equation and the specified z and c values
    int evaluate (std::complex<long double>, int); // Perform the fractal calculation
```

```
37
38      HFractalEquation (std::vector<Token>); // Initialise with a sequence of
    equation tokens
39      HFractalEquation (); // Base initialiser
40  };
41
42  #endif
```

```cpp
// src/gui.cc

#include "gui.hh"
#include "guimain.hh"

#include <math.h>
#include <algorithm>
#include <thread>

#include "utils.hh"
#include "database.hh"

using namespace std;

/**
 * @brief Automatically configure the styling for the GUI.
 * Uses a stylesheet provided by raysan5, creator of the graphics library used in the project, raylib
 *
 */
void HFractalGui::configureStyling() {
    // This function implements the 'cyber' interface style provided by raygui's documentation.
    const char* stylesheet = R"(p 00 00 0x2f7486ff DEFAULT_BORDER_COLOR_NORMAL
p 00 01 0x024658ff    DEFAULT_BASE_COLOR_NORMAL
p 00 02 0x51bfd3ff    DEFAULT_TEXT_COLOR_NORMAL
p 00 03 0x82cde0ff    DEFAULT_BORDER_COLOR_FOCUSED
p 00 04 0x3299b4ff    DEFAULT_BASE_COLOR_FOCUSED
p 00 05 0xb6e1eaff    DEFAULT_TEXT_COLOR_FOCUSED
p 00 06 0xeb7630ff    DEFAULT_BORDER_COLOR_PRESSED
p 00 07 0xffbc51ff    DEFAULT_BASE_COLOR_PRESSED
p 00 08 0xd86f36ff    DEFAULT_TEXT_COLOR_PRESSED
p 00 09 0x134b5aff    DEFAULT_BORDER_COLOR_DISABLED
p 00 10 0x02313dff    DEFAULT_BASE_COLOR_DISABLED
p 00 11 0x17505fff    DEFAULT_TEXT_COLOR_DISABLED
p 00 16 0x00000012    DEFAULT_TEXT_SIZE
p 00 17 0x00000001    DEFAULT_TEXT_SPACING
p 00 18 0x81c0d0ff    DEFAULT_LINE_COLOR
p 00 19 0x00222bff    DEFAULT_BACKGROUND_COLOR)";
    // Iterate over string and extract styling properties
    int offset = 0;
```

```
40      int stylePointIndex = 0;
41      string stylePointControl = "";
42      string stylePointProperty = "";
43      string stylePointValue = "";
44      while (offset <= strlen(stylesheet)) {
45          if (stylesheet[offset] == ' ') {
46              stylePointIndex++;
47          } else if (stylesheet[offset] == '\n' || stylesheet[offset] ==
'\0') {
48              GuiSetStyle (stoi(stylePointControl), stoi(stylePointProperty),
stoll(stylePointValue, nullptr, 16));
49              stylePointControl = "";
50              stylePointProperty = "";
51              stylePointValue = "";
52              stylePointIndex = 0;
53          } else {
54              switch (stylePointIndex) {
55              case 1:
56                  stylePointControl += stylesheet[offset];
57                  break;
58              case 2:
59                  stylePointProperty += stylesheet[offset];
60                  break;
61              case 3:
62                  stylePointValue += stylesheet[offset];
63                  break;
64              default:
65                  break;
66              }
67          }
68          offset++;
69      }
70      // Tell raylib we've finished updating the styling
71      GuiUpdateStyleComplete();
72  }
73
74  /**
75   * @brief Configure the GUI itself and all class properties
76   *
77   */
78  void HFractalGui::configureGUI(char* path) {
79      // Basic class initialisation
```

```
80      dialog_text = "";
81      console_text = "Ready.";
82      save_name_buffer = "Untitled";
83      for (int i = 0; i < BUTTON_NUM_TOTAL; i++) button_states[i] = false;
84      buffer_image = {};
85      buffer_texture = {};
86      is_rendering = false;
87      is_outdated_render = true;
88      render_percentage = 0;
89      showing_coordinates = false;
90      modal_view_state = MVS_NORMAL;
91      selected_palette = CP_RAINBOW;
92      database_load_dialog_scroll = 0;
93      textbox_focus = TEXT_FOCUS_STATE::TFS_NONE;
94
95      // Fetch configuration
96      unsigned int thread_count = std::thread::hardware_concurrency ();
97      long double start_zoom = 1;
98      long double start_x_offset = 0;
99      long double start_y_offset = 0;
100     image_dimension = WINDOW_INIT_HEIGHT;
101     control_panel_width = WINDOW_INIT_WIDTH - WINDOW_INIT_HEIGHT;
102     equation_buffer = equationPreset (EQ_MANDELBROT, false);
103
104     // GUI configuration
105     SetTraceLogLevel (LOG_WARNING | LOG_ERROR | LOG_DEBUG);
106     InitWindow(WINDOW_INIT_WIDTH, WINDOW_INIT_HEIGHT, "HyperFractal
Mathematical Visualiser");
107     SetWindowState (FLAG_WINDOW_RESIZABLE);
108     SetExitKey(-1);
109     int min_height = std::max (256, CONTROL_MIN_HEIGHT);
110     SetWindowMinSize(min_height+CONTROL_MIN_WIDTH, min_height);
111     SetTargetFPS(24);
112     configureStyling();
113
114     // Initialise database;
115     database = HFractalDatabase
(string(path)+string("FractalSavedStates.csv"));
116
117     // Initialise rendering environment
118     lowres_hm = new HFractalMain();
119     hm = new HFractalMain();
```

```
120
121     // Configure full resolution renderer
122     hm->setResolution (image_dimension);
123     hm->setEquation (equation_buffer);
124     hm->setEvalLimit (200);
125     hm->setWorkerThreads (thread_count);
126     hm->setZoom (start_zoom);
127     hm->setOffsetX (start_x_offset);
128     hm->setOffsetY (start_y_offset);
129
130     // Configure preivew renderer
131     lowres_hm->setResolution (128);
132     lowres_hm->setEquation (equation_buffer);
133     lowres_hm->setEvalLimit (200);
134     lowres_hm->setWorkerThreads (thread_count/2);
135     lowres_hm->setZoom (start_zoom);
136     lowres_hm->setOffsetX (start_x_offset);
137     lowres_hm->setOffsetY (start_y_offset);
138 }
139
140 /**
141  * @brief Called when a rendering parameter has been modified.
142  * Causes the rendered image to become 'out of date' and updates the
    preview render
143  *
144  */
145 void HFractalGui::parametersWereModified() {
146     is_outdated_render = true;
147     console_text = "Outdated render!";
148     updatePreviewRender();
149 }
150
151 /**
152  * @brief Generate and show an updated image from the preview renderer
153  *
154  * @return True if the update was successful, false if the equation was
    invalid
155  */
156 bool HFractalGui::updatePreviewRender() {
157     // Check if the equation is valid
158     if (!lowres_hm->isValidEquation()) return false;
159     lowres_hm->generateImage(true); // If it is, run a render
```

```
160     reloadImageFrom(lowres_hm); // And load it
161     return true;
162 }
163 
164 /**
165  * @brief Trigger a full resolution render to start
166  *
167  * @return True if successful, false if the equation was invalid
168  */
169 bool HFractalGui::startFullRender() {
170     if (!hm->isValidEquation()) { // Check if this is a valid equation
171         console_text = "Invalid equation!";
172         return false;
173     }
174     // If it is, start the render
175     is_rendering = true;
176     console_text = "Rendering...";
177     hm->generateImage(false);
178     is_outdated_render = true;
179     render_percentage = 0;
180     return true;
181 }
182 
183 /**
184  * @brief Check the status of the full resolution render, and produce an
    up-to-date display image showing the render progress.
185  * Also updates the completion percentage
186  *
187  * @return True if the image has finished rendering, false otherwise
188  */
189 bool HFractalGui::updateFullRender() {
190     if (!is_rendering) return true;
191     // Update completion percentage
192     render_percentage = round(hm->getImageCompletionPercentage());
193     if (hm->getIsRendering()) { // If still rendering, update the rendered
    image and overlay it onto the preview
194         is_outdated_render = true;
195         Image overlay = getImage(hm);
196         ImageDraw(&buffer_image, overlay, (Rectangle){0,0,(float)hm-
    >getResolution(),(float)hm->getResolution()}, (Rectangle){0,0,(float)hm-
    >getResolution(),(float)hm->getResolution()}, WHITE);
197         UnloadImage(overlay);
198         tryUnloadTexture();
```

```
199         buffer_texture = LoadTextureFromImage(buffer_image);
200         return false;
201     } else { // Otherwise, set states to indicate completion and update the
    image
202         is_outdated_render = false;
203         is_rendering = false;
204         reloadImageFrom (hm);
205         console_text = "Rendering done.";
206         return true;
207     }
208 }
209 
210 /**
211  * @brief Reload the image and texture used for drawing to the screen from
    the specified render environment
212  *
213  * @param h Rendering environment to grab the image from
214  */
215 void HFractalGui::reloadImageFrom(HFractalMain* h) {
216     tryUnloadImage(); // Unload image and texture
217     tryUnloadTexture();
218     buffer_image = getImage(h);
219     if (buffer_image.height != image_dimension) // Resize it to fill the
    frame
220         ImageResize(&buffer_image, image_dimension, image_dimension);
221     buffer_texture = LoadTextureFromImage(buffer_image);
222 }
223 
224 /**
225  * @brief Check if the window has been resized, and handle it if so
226  *
227  */
228 void HFractalGui::checkWindowResize() {
229     if (is_rendering) { // If we're mid-render, snap back to previous
    window dimensions
230         SetWindowSize(image_dimension+control_panel_width,
    image_dimension);
231         return;
232     }
233     if (IsWindowResized()) { // Update render resolution and image
    dimension based on new size
234         image_dimension = std::min(GetScreenWidth()-CONTROL_MIN_WIDTH,
    GetScreenHeight());
235         control_panel_width = GetScreenWidth()-image_dimension;
```

```
236         hm->setResolution(image_dimension);
237         parametersWereModified(); // Notify that parameters have changed
238     }
239 }
240 
241 /**
242  * @brief Draw the entire interface
243  *
244  */
245 void HFractalGui::drawInterface() {
246     BeginDrawing(); // Tell raylib we're about to start drawing
247 
248     // Clear the background
249     Color bg_col = GetColor (GuiGetStyle(00, BACKGROUND_COLOR));
250     ClearBackground(bg_col);
251 
252     // Draw the rendered HFractalImage
253     Vector2 v {0,0};
254     DrawTextureEx (buffer_texture, v, 0,
(float)image_dimension/(float)buffer_texture.height, WHITE);
255     // Draw Console
256     int button_offset = 0;
257     GuiTextBox((Rectangle){(float)image_dimension, BUTTON_HEIGHT*
(float)button_offset, (float)control_panel_width, BUTTON_HEIGHT},
(char*)console_text.c_str(), 1, false);
258     // Draw "Render Image" button
259     button_offset++;
260     button_states[BUTTON_ID::BUTTON_ID_RENDER] = GuiButton((Rectangle)
{(float)image_dimension, BUTTON_HEIGHT*(float)button_offset,
(float)control_panel_width, BUTTON_HEIGHT}, "Render Image") &&
(modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
261     // Draw render progress bar
262     button_offset++;
263     GuiProgressBar ((Rectangle){(float)image_dimension, BUTTON_HEIGHT*
(float)button_offset, (float)control_panel_width, BUTTON_HEIGHT}, "", "",
render_percentage, 0, 100);
264     // Draw zoom buttons
265     button_offset++;
266     button_states[BUTTON_ID::BUTTON_ID_ZOOM_IN] = GuiButton((Rectangle)
{(float)image_dimension, BUTTON_HEIGHT*(float)button_offset,
(float)control_panel_width/3, BUTTON_HEIGHT}, "Zoom In") &&
(modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
267     button_states[BUTTON_ID::BUTTON_ID_ZOOM_RESET] = GuiButton((Rectangle)
{(float)image_dimension+(float)control_panel_width/3, BUTTON_HEIGHT*
(float)button_offset, (float)control_panel_width/3, BUTTON_HEIGHT}, "Reset
Zoom") && (modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
```

```
268     button_states[BUTTON_ID::BUTTON_ID_ZOOM_OUT] = GuiButton((Rectangle)
{(float)image_dimension+(float)control_panel_width/(3.0f/2.0f),
BUTTON_HEIGHT*(float)button_offset, (float)control_panel_width/3,
BUTTON_HEIGHT}, "Zoom Out") && (modal_view_state ==
MODAL_VIEW_STATE::MVS_NORMAL);
269     // Draw save image button
270     button_offset++;
271     button_states[BUTTON_ID::BUTTON_ID_SAVE_IMAGE] = GuiButton((Rectangle)
{(float)image_dimension, BUTTON_HEIGHT*(float)button_offset,
(float)control_panel_width, BUTTON_HEIGHT}, "Save Image") &&
(modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
272     // Draw render state load/save buttons
273     button_offset++;
274     button_states[BUTTON_ID::BUTTON_ID_SAVE_RSTATE] = GuiButton((Rectangle)
{(float)image_dimension, BUTTON_HEIGHT*(float)button_offset,
(float)control_panel_width/2, BUTTON_HEIGHT}, "Save Render State") &&
(modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
275     button_states[BUTTON_ID::BUTTON_ID_LOAD_RSTATE] = GuiButton((Rectangle)
{(float)image_dimension+(float)control_panel_width/2, BUTTON_HEIGHT*
(float)button_offset, (float)control_panel_width/2, BUTTON_HEIGHT}, "Load
Render State") && (modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
276     // Draw movement navigation buttons
277     button_offset++;
278     button_states[BUTTON_ID::BUTTON_ID_UP] = GuiButton((Rectangle)
{(float)image_dimension+((float)control_panel_width-40)/2, BUTTON_HEIGHT*
(float)button_offset, (float)40, BUTTON_HEIGHT}, "up") && (modal_view_state
== MODAL_VIEW_STATE::MVS_NORMAL);
279     button_offset++;
280     button_states[BUTTON_ID::BUTTON_ID_LEFT] = GuiButton((Rectangle)
{(float)image_dimension+(((float)control_panel_width)/2)-40, BUTTON_HEIGHT*
(float)button_offset, (float)40, BUTTON_HEIGHT}, "left") &&
(modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
281     button_states[BUTTON_ID::BUTTON_ID_RIGHT] = GuiButton((Rectangle)
{(float)image_dimension+(((float)control_panel_width)/2), BUTTON_HEIGHT*
(float)button_offset, (float)40, BUTTON_HEIGHT}, "right") &&
(modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
282     button_offset++;
283     button_states[BUTTON_ID::BUTTON_ID_DOWN] = GuiButton((Rectangle)
{(float)image_dimension+((float)control_panel_width-40)/2, BUTTON_HEIGHT*
(float)button_offset, (float)40, BUTTON_HEIGHT}, "down") &&
(modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
284     button_offset++;
285     button_states[BUTTON_ID::BUTTON_ID_EQ_PRESETS] = GuiButton((Rectangle)
{(float)image_dimension+(float)control_panel_width/2, BUTTON_HEIGHT*
(float)button_offset, (float)control_panel_width/2, BUTTON_HEIGHT},
"Equation Presets") && (modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
286
287     // Draw equation input box
288     button_states[BUTTON_ID::BUTTON_ID_EQ_INPUTBOX] = GuiTextBox
((Rectangle){(float)image_dimension, BUTTON_HEIGHT*(float)button_offset,
(float)control_panel_width/2, BUTTON_HEIGHT}, equation_buffer.data(), 1,
false) && (modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
```

```
289     button_offset++;
290 
291     // Coordinate toggle button
292     string coord_button_text = "Hide coordinates";
293     if (!showing_coordinates) coord_button_text = "Show coordinates";
294     button_states[BUTTON_ID::BUTTON_ID_TOGGLE_COORDS] =
GuiButton((Rectangle){(float)image_dimension, BUTTON_HEIGHT*
(float)button_offset, (float)control_panel_width, BUTTON_HEIGHT},
coord_button_text.c_str()) && (modal_view_state ==
MODAL_VIEW_STATE::MVS_NORMAL);
295     button_offset++;
296 
297     // Eval limit controls
298     button_states[BUTTON_ID::BUTTON_ID_EVAL_LIM_LESS] =
GuiButton((Rectangle){(float)image_dimension+(float)control_panel_width/2,
BUTTON_HEIGHT*(float)button_offset, (float)control_panel_width/4,
BUTTON_HEIGHT}, "<") && (modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
299     button_states[BUTTON_ID::BUTTON_ID_EVAL_LIM_MORE] =
GuiButton((Rectangle){(float)image_dimension+
((float)control_panel_width/4)*3, BUTTON_HEIGHT*(float)button_offset,
(float)control_panel_width/4, BUTTON_HEIGHT}, ">") && (modal_view_state ==
MODAL_VIEW_STATE::MVS_NORMAL);
300     char evalLimString[16];
301     sprintf (evalLimString, "%d (%d)", hm->getEvalLimit(), lowres_hm-
>getEvalLimit());
302     GuiTextBox ((Rectangle){(float)image_dimension, BUTTON_HEIGHT*
(float)button_offset, (float)control_panel_width/2, BUTTON_HEIGHT},
evalLimString, 1, false);
303     button_offset++;
304 
305     // Colour palette preset selector button
306     button_states[BUTTON_ID::BUTTON_ID_CP_PRESETS] = GuiButton((Rectangle)
{(float)image_dimension, BUTTON_HEIGHT*(float)button_offset,
(float)control_panel_width, BUTTON_HEIGHT}, "Colour Palettes") &&
(modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
307 
308     // Draw help button
309     button_states[BUTTON_ID::BUTTON_ID_HELP] = GuiButton((Rectangle)
{(float)image_dimension, (float)GetScreenHeight()-(2*BUTTON_HEIGHT),
(float)control_panel_width, BUTTON_HEIGHT*2}, "Help & Instructions") &&
(modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL);
310 
311     // Draw the equation preset dialog
312     if (modal_view_state == MODAL_VIEW_STATE::MVS_EQUATION_PRESET_SELECTOR)
{
313         float preset_dialog_x = (float)image_dimension+
(float)control_panel_width/2;
314         float preset_dialog_y = BUTTON_HEIGHT*10.0f;
315         for (int e = 0; e < NUM_EQUATION_PRESETS; e++) {
```

```
316             // Draw a button for each option
317             if (
318                 GuiButton((Rectangle){preset_dialog_x, preset_dialog_y+
(BUTTON_HEIGHT*e), (float)control_panel_width/2, BUTTON_HEIGHT},
equationPreset((EQ_PRESETS)e, true).c_str())
319             && !is_rendering
320             ) {
321                 escapeEquationPresetDialog(e);
322             }
323         }
324         if (GetMouseX() < preset_dialog_x || GetMouseX() > preset_dialog_x
+ (float)control_panel_width/2 ||  GetMouseY() < preset_dialog_y -
BUTTON_HEIGHT || GetMouseY() > preset_dialog_y +
(BUTTON_HEIGHT*NUM_EQUATION_PRESETS)) {
325             escapeEquationPresetDialog(-1);
326         }
327     }
328
329     // Draw the colour palette preset dialog
330     if (modal_view_state == MODAL_VIEW_STATE::MVS_COLOUR_PRESET_SELECTOR) {
331         float preset_dialog_x = (float)image_dimension;
332         float preset_dialog_y = BUTTON_HEIGHT*13.0f;
333         for (int c = 0; c < NUM_COLOUR_PRESETS; c++) {
334             // Draw a button for each option
335             if (
336                 GuiButton((Rectangle){preset_dialog_x, preset_dialog_y+
(BUTTON_HEIGHT*c), (float)control_panel_width, BUTTON_HEIGHT},
colourPalettePreset((CP_PRESETS)c).c_str())
337             && !is_rendering
338             ) {
339                 escapeColourPalettePresetDialog(c);
340             }
341         }
342         if (GetMouseX() < preset_dialog_x || GetMouseX() > preset_dialog_x
+ (float)control_panel_width ||  GetMouseY() < preset_dialog_y -
BUTTON_HEIGHT || GetMouseY() > preset_dialog_y +
(BUTTON_HEIGHT*NUM_COLOUR_PRESETS)) {
343             escapeColourPalettePresetDialog(-1);
344         }
345     }
346
347     // Draw the info dialog
348     if (modal_view_state == MODAL_VIEW_STATE::MVS_TEXT_DIALOG) {
349         float box_width = (2.0/3.0)*GetScreenWidth();
350         DrawRectangle (0, 0, GetScreenWidth(), GetScreenHeight(), (Color)
```

```
{200, 200, 200, 128});
351         Rectangle text_rec = (Rectangle){
352             ((float)GetScreenWidth()-box_width-10)/2,
353             ((float)GetScreenHeight()-DIALOG_TEXT_SIZE-10)/2,
354             box_width+10,
355             DIALOG_TEXT_SIZE
356         };
357         GuiDrawText (dialog_text.c_str(), text_rec, GUI_TEXT_ALIGN_CENTER,
BLACK);
358         button_states[BUTTON_ID::BUTTON_ID_TEXT_DIALOG_CLOSE] =
GuiButton((Rectangle){(float)(GetScreenWidth()-box_width-10)/2, (float)
(GetScreenHeight()*(3.0/4.0)+10), (float)(box_width+10), (float)
(DIALOG_TEXT_SIZE+10)}, "OK");
359     }
360
361     // Draw database dialog
362     if (modal_view_state == MODAL_VIEW_STATE::MVS_DATABASE_SAVE_DIALOG ||
modal_view_state == MODAL_VIEW_STATE::MVS_DATABASE_LOAD_DIALOG) {
363         DrawRectangle (0, 0, GetScreenWidth(), GetScreenHeight(), (Color)
{200, 200, 200, 128});
364         float box_width = (2.0/3.0)*GetScreenWidth();
365         button_states[BUTTON_ID::BUTTON_ID_DATABASE_CANCEL] = GuiButton(
366             (Rectangle){
367                 (float)(GetScreenWidth()-box_width-10)/2,
368                 (float)(GetScreenHeight()*(4.0/5.0)+10),
369                 (float)((box_width+10)/2.0),
370                 (float)(DIALOG_TEXT_SIZE+10)
371                 },
372         "Cancel");
373
374         // Branch depending on whether the saving dialog or the loading
dialog is open
375         if (modal_view_state == MODAL_VIEW_STATE::MVS_DATABASE_SAVE_DIALOG)
{
376             button_states[BUTTON_ID::BUTTON_ID_SAVE] = GuiButton(
377                 (Rectangle){
378                     (float)(GetScreenWidth()-10)/2,
379                     (float)(GetScreenHeight()*(4.0/5.0)+10),
380                     (float)((box_width+10)/2.0),
381                     (float)(DIALOG_TEXT_SIZE+10)
382                     },
383             "Save");
384
385             button_states[BUTTON_ID::BUTTON_ID_SAVE_NAME_INPUTBOX] =
GuiTextBox (
```

```
                (Rectangle){
                    (float)((GetScreenWidth()-box_width)/2.0),
                    (float)(GetScreenHeight()*(1.0/5.0)),
                    (float)(box_width),
                    (float)(BUTTON_HEIGHT*2)
                    },
            save_name_buffer.data(), 2, false);

        } else if (modal_view_state == MODAL_VIEW_STATE::MVS_DATABASE_LOAD_DIALOG) {
            button_states[BUTTON_ID::BUTTON_ID_LOAD] = GuiButton(
                (Rectangle){
                    (float)(GetScreenWidth()-10)/2,
                    (float)(GetScreenHeight()*(4.0/5.0)+10),
                    (float)((box_width+10)/2.0),
                    (float)(DIALOG_TEXT_SIZE+10)
                    },
            "Load (overwrites current config)");

            button_states[BUTTON_ID::BUTTON_ID_SCROLL_UP] = GuiButton(
                (Rectangle){
                    (float)(GetScreenWidth()/2),
                    (float)(GetScreenHeight()*(1.0/5.0))+9*BUTTON_HEIGHT,
                    (float)120,
                    (float)(BUTTON_HEIGHT)
                    },
            "Scroll up");

            button_states[BUTTON_ID::BUTTON_ID_SCROLL_DOWN] = GuiButton(
                (Rectangle){
                    (float)(GetScreenWidth()/2),
                    (float)(GetScreenHeight()*(1.0/5.0))+10*BUTTON_HEIGHT,
                    (float)120,
                    (float)(BUTTON_HEIGHT)
                    },
            "Scroll down");

            auto descriptions = database.getConfigDescriptions();
            int row_offset = 0;

            for (auto item : descriptions) {
                int draw_row = row_offset-database_load_dialog_scroll;
```

```
                if (draw_row >= 0 && draw_row <= 8) {
                    if (
                        GuiButton(
                    (Rectangle){
                        (float)(GetScreenWidth()-box_width)/2,
                        (float)(GetScreenHeight()*(1.0/5.0) + BUTTON_HEIGHT*draw_row),
                        (float)((box_width)-120),
                        (float)(BUTTON_HEIGHT)
                        },
                    (((item.first == selected_profile_id) ? "(x)" : "( )") + item.second).c_str())
                    ) {
                        selected_profile_id = item.first;
                    }
                    if (
                        GuiButton(
                    (Rectangle){
                        (float)((GetScreenWidth()+box_width)/2)-120,
                        (float)(GetScreenHeight()*(1.0/5.0) + BUTTON_HEIGHT*draw_row),
                        (float)(120),
                        (float)(BUTTON_HEIGHT)
                        },
                    "Delete? (!)")
                    ) {
                        database.removeConfig(item.first);
                        database.commit();
                    }
                }
                row_offset++;
            }
        }
    }

    // Draw coordinates text next to cursor
    if (modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL && showing_coordinates && GetMouseX() <= image_dimension && GetMouseY() <= image_dimension) {
        float left = GetMouseX()+15;
        float top = GetMouseY()+15;
        Color col {250, 250, 250, 200};

```

```
465         long double location_x = hm->getOffsetX() + ((long double)(((long double)GetMouseX()/(image_dimension/2))-1))/hm->getZoom();
466         long double location_y = hm->getOffsetY() - ((long double)(((long double)GetMouseY()/(image_dimension/2))-1))/hm->getZoom();
467         char t[142];
468         sprintf (t, "%.10Lf\n%.10Lf", location_x, location_y);
469         DrawRectangle (left, top, 115, 40, col);
470         DrawText (t, left+5, top, 15, BLACK);
471     }
472 
473     EndDrawing(); // Tell raylib we're done drawing
474 }
475 
476 /**
477  * @brief Close the equation preset dialog, and switch to a given preset
478  *
479  * @param e The equation preset to switch to, or -1 if the dialog was cancelled
480  */
481 void HFractalGui::escapeEquationPresetDialog(int e) {
482     modal_view_state = MODAL_VIEW_STATE::MVS_NORMAL; // Switch back to normal mode
483     if (is_rendering) return;
484     if (e != -1) { // If an option was selected, make it the current equation and notify that parameters have changed
485         equation_buffer = equationPreset ((EQ_PRESETS)e, false);
486         hm->setEquation (equation_buffer);
487         lowres_hm->setEquation (equation_buffer);
488         // Check whether the equation is valid
489         if (!hm->isValidEquation()) console_text = "Invalid equation input";
490         else {
491             parametersWereModified();
492         }
493     }
494 }
495 
496 /**
497  * @brief Show the equation preset dialog
498  *
499  */
500 void HFractalGui::enterEquationPresetDialog() {
501     if (is_rendering) return;
502     // Switch to equation preset selector mode
```

```
503     modal_view_state = MODAL_VIEW_STATE::MVS_EQUATION_PRESET_SELECTOR;
504 }
505 
506 /**
507  * @brief Show the colour palette preset dialog
508  *
509  */
510 void HFractalGui::enterColourPalettePresetDialog() {
511     if (is_rendering) return;
512     // Switch to colour preset selector mode
513     modal_view_state = MODAL_VIEW_STATE::MVS_COLOUR_PRESET_SELECTOR;
514 }
515 
516 /**
517  * @brief Close the colour palette preset dialog and switch to a given
    palette
518  *
519  * @param c Palette to switch to, or -1 if the dialog was cancelled
520  */
521 void HFractalGui::escapeColourPalettePresetDialog(int c) {
522     modal_view_state = MODAL_VIEW_STATE::MVS_NORMAL; // Return to normal
    GUI mode
523     if (is_rendering) return;
524     if (c != -1) {
525         // If an option was selected, reload the image with the selected
    palette (no rerender necessarry)
526         selected_palette = (CP_PRESETS)c;
527         if (is_outdated_render) {
528             reloadImageFrom(lowres_hm);
529         } else {
530             reloadImageFrom(hm);
531         }
532     }
533 }
534 
535 /**
536  * @brief Get an image handleable by raylib from a given rendering
    environment
537  *
538  * @param h Rendering environment to extract from
539  * @return A raylib-style image for drawing into the GUI
540  */
541 Image HFractalGui::getImage(HFractalMain* h) {
```

```
542     // Fetch a 32 bit RGBA image in the selected colour palette
543     int size = h->getResolution();
544     uint32_t *data = h->getRGBAImage(selected_palette);
545     Color *pixels = (Color *)malloc (size*size*sizeof(Color));
546     // Convert the image data to a format raylib will accept
547     for (int i = 0; i < size*size; i++) pixels[i] = GetColor(data[i]);
548     delete data;
549     // Construct a raylib image from the data
550     Image img = {
551         .data = pixels,
552         .width = size,
553         .height = size,
554         .mipmaps = 1,
555         .format = PIXELFORMAT_UNCOMPRESSED_R8G8B8A8
556     };
557     return img;
558 }
559 
560 /**
561  * @brief Handle when the user clicks on the image.
562  * Automatically centres the area they clicked and notifies the GUI that
    rendering parameters have been modified, triggering a preview update
563  *
564  * @return True if a click was handled, otherwise false
565  */
566 bool HFractalGui::handleClickNavigation() {
567     if (IsMouseButtonPressed(MOUSE_LEFT_BUTTON) && !is_rendering) {
568         Vector2 mpos = GetMousePosition();
569         // Check if the mouse click was inside the image
570         if (mpos.x <= image_dimension && mpos.y <= image_dimension) {
571             long double change_in_x = (long double)((mpos.x /
    (image_dimension / 2)) - 1) / hm->getZoom();
572             long double change_in_y = (long double)((mpos.y /
    (image_dimension / 2)) - 1) / hm->getZoom();
573             long double new_offset_x = hm->getOffsetX() + change_in_x;
574             long double new_offset_y = hm->getOffsetY() - change_in_y;
575             // Update parameters and notify of the modification
576             lowres_hm->setOffsetX(new_offset_x);
577             lowres_hm->setOffsetY(new_offset_y);
578             hm->setOffsetX(new_offset_x);
579             hm->setOffsetY(new_offset_y);
580             parametersWereModified();
581             return true;
```

```
582         }
583     }
584     return false;
585 }
586 
587 /**
588  * @brief Show a text dialog with a given string as text
589  *
590  * @param text Text to display
591  */
592 void HFractalGui::launchTextDialog(std::string text) {
593     modal_view_state = MODAL_VIEW_STATE::MVS_TEXT_DIALOG;
594     dialog_text = text;
595 }
596 
597 /**
598  * @brief Close the currently open text dialog and go back to normal GUI
    mode
599  *
600  */
601 void HFractalGui::closeTextDialog() {
602     modal_view_state = MODAL_VIEW_STATE::MVS_NORMAL;
603     dialog_text = "";
604 }
605 
606 /**
607  * @brief Handler for Zoom In button
608  *
609  */
610 void HFractalGui::zoomIn() {
611     if (hm->getZoom() <= SCALE_DEPTH_LIMIT) { // Check the zoom has not
    exceeded the depth limit
612         long double new_zoom = hm->getZoom() * SCALE_STEP_FACTOR;
613         lowres_hm->setZoom (new_zoom);
614         hm->setZoom (new_zoom);
615         parametersWereModified();
616     } else launchTextDialog ("Zoom precision limit reached"); // Present a
    text dialog to report the issue to the user
617 }
618 
619 /**
620  * @brief Handler for Zoom Out button
621  *
```

```
622  */
623 void HFractalGui::zoomOut() {
624     long double new_zoom = hm->getZoom() / SCALE_STEP_FACTOR;
625     lowres_hm->setZoom (new_zoom);
626     hm->setZoom (new_zoom);
627     parametersWereModified();
628 }
629 
630 /**
631  * @brief Handler for Reset Zoom button
632  *
633  */
634 void HFractalGui::resetZoom() {
635     lowres_hm->setZoom(1);
636     hm->setZoom(1);
637     parametersWereModified();
638 }
639 
640 /**
641  * @brief Handler for Save Image button
642  *
643  */
644 void HFractalGui::saveImage() {
645     bool result = false;
646     // Switch depending on whether there is a full render available to save
647     if (is_outdated_render) {
648         result = lowres_hm->autoWriteImage(IMAGE_TYPE::PGM);
649         console_text = "Saved preview render to desktop.";
650     } else {
651         result = hm->autoWriteImage (IMAGE_TYPE::PGM);
652         console_text = "Saved render to desktop.";
653     }
654     if (!result) {
655         console_text = "Image saving failed.";
656     }
657 }
658 
659 /**
660  * @brief Make the save render state dialog visible
661  *
662  */
```

```
663 void HFractalGui::showSaveStateDialog() {
664     if (is_rendering) return;
665     modal_view_state = MODAL_VIEW_STATE::MVS_DATABASE_SAVE_DIALOG;
666 }
667 
668 /**
669  * @brief Make the load render state dialog visible
670  *
671  */
672 void HFractalGui::showLoadStateDialog() {
673     if (is_rendering) return;
674     modal_view_state = MODAL_VIEW_STATE::MVS_DATABASE_LOAD_DIALOG;
675     database_load_dialog_scroll = 0;
676 }
677 
678 /**
679  * @brief Save the current render state to the database and close the
    dialog
680  *
681  */
682 void HFractalGui::saveStateToDatabase() {
683     // Create the new config profile and populate its fields
684     HFractalConfigProfile *cp = new HFractalConfigProfile();
685     cp->equation = hm->getEquation();
686     cp->iterations = hm->getEvalLimit();
687     cp->name = save_name_buffer;
688     cp->palette = selected_palette;
689     cp->x_offset = hm->getOffsetX();
690     cp->y_offset = hm->getOffsetY();
691     cp->zoom = hm->getZoom();
692 
693     // Fetch or create the default user if necessarry
694     HFractalUserProfile *default_user = database.getUser(0);
695     if (default_user == NULL) {
696         default_user = new HFractalUserProfile();
697         default_user->user_name = "default";
698         database.insertUser (default_user);
699     }
700 
701     cp->user_id = default_user->user_id;
702 
703     // Insert the new profile into the database
```

```
704     database.insertConfig(cp);
705     database.commit();
706     console_text = "Profile saved to database!";
707     closeDatabaseDialog(); // Escape from the dialog
708 }
709 
710 /**
711  * @brief Load the selected render state from the database and close the
    dialog
712  *
713  */
714 void HFractalGui::loadStateFromDatabase() {
715     // Try to fetch the config profile
716     HFractalConfigProfile *cp = database.getConfig (selected_profile_id);
717     if (cp == NULL) {
718         console_text = "No profile selected to load.";
719     } else { // On success, load all properties into the rendering
    environments
720         hm->setEquation(cp->equation);
721         lowres_hm->setEquation(cp->equation);
722 
723         hm->setEvalLimit(cp->iterations);
724         lowres_hm->setEvalLimit(cp->iterations);
725 
726         hm->setOffsetX(cp->x_offset);
727         lowres_hm->setOffsetX(cp->x_offset);
728 
729         hm->setOffsetY(cp->y_offset);
730         lowres_hm->setOffsetY(cp->y_offset);
731 
732         hm->setZoom(cp->zoom);
733         lowres_hm->setZoom(cp->zoom);
734 
735         selected_palette = (CP_PRESETS)cp->palette;
736         save_name_buffer = cp->name;
737 
738         updatePreviewRender();
739         console_text = "Profile '" + save_name_buffer + "' loaded from
    database.";
740     }
741     closeDatabaseDialog(); // Close the dialog
742 }
743 
```

```
744 /**
745  * @brief Hide the database dialog and return to normal GUI mode
746  *
747  */
748 void HFractalGui::closeDatabaseDialog() {
749     modal_view_state = MODAL_VIEW_STATE::MVS_NORMAL;
750 }
751
752 /**
753  * @brief Scroll down inside the load render state dialog
754  *
755  */
756 void HFractalGui::databaseLoadScrollDown() {
757     database_load_dialog_scroll++;
758 }
759
760 /**
761  * @brief Scroll up inside the load render state dialog
762  *
763  */
764 void HFractalGui::databaseLoadScrollUp() {
765     database_load_dialog_scroll--;
766     if (database_load_dialog_scroll < 0) database_load_dialog_scroll = 0;
767 }
768
769 /**
770  * @brief Handler for Move Up button
771  *
772  */
773 void HFractalGui::moveUp() {
774     long double new_offset = hm->getOffsetY() + (MOVE_STEP_FACTOR/hm-
>getZoom());
775     hm->setOffsetY (new_offset);
776     lowres_hm->setOffsetY (new_offset);
777     parametersWereModified();
778 }
779
780 /**
781  * @brief Handler for Move Left button
782  *
783  */
784 void HFractalGui::moveLeft() {
```

```
785     long double new_offset = hm->getOffsetX() - (MOVE_STEP_FACTOR/hm-
>getZoom());
786     hm->setOffsetX (new_offset);
787     lowres_hm->setOffsetX (new_offset);
788     parametersWereModified();
789 }
790 
791 /**
792  * @brief Handler for Move Right button
793  *
794  */
795 void HFractalGui::moveRight() {
796     long double new_offset = hm->getOffsetX() + (MOVE_STEP_FACTOR/hm-
>getZoom());
797     hm->setOffsetX (new_offset);
798     lowres_hm->setOffsetX (new_offset);
799     parametersWereModified();
800 }
801 
802 /**
803  * @brief Handler for Move Down button
804  *
805  */
806 void HFractalGui::moveDown() {
807     long double new_offset = hm->getOffsetY() - (MOVE_STEP_FACTOR/hm-
>getZoom());
808     hm->setOffsetY (new_offset);
809     lowres_hm->setOffsetY (new_offset);
810     parametersWereModified();
811 }
812 
813 /**
814  * @brief Handler for Show/Hide Coordinates button
815  *
816  */
817 void HFractalGui::toggleCoords() {
818     showing_coordinates = !showing_coordinates;
819 }
820 
821 /**
822  * @brief Handlder for '<' button
823  *
824  */
```

```
825 void HFractalGui::evalLimitLess() {
826     int new_el = hm->getEvalLimit();
827     // Allow faster jumping if shift is held
828     if (IsKeyDown(KEY_LEFT_SHIFT) || IsKeyDown (KEY_RIGHT_SHIFT)) {
829         new_el -= 10;
830     } else {
831         new_el--;
832     }
833     hm->setEvalLimit (new_el);
834     lowres_hm->setEvalLimit (new_el);
835     parametersWereModified();
836 }
837 
838 /**
839  * @brief Handler for '>' button
840  *
841  */
842 void HFractalGui::evalLimitMore() {
843     int new_el = hm->getEvalLimit();
844     // Allow faster jumping if shift is held
845     if (IsKeyDown(KEY_LEFT_SHIFT) || IsKeyDown (KEY_RIGHT_SHIFT)) {
846         new_el += 10;
847     } else {
848         new_el++;
849     }
850     hm->setEvalLimit (new_el);
851     lowres_hm->setEvalLimit (new_el);
852     parametersWereModified();
853 }
854 
855 /**
856  * @brief Handler for Help & Instructions button
857  *
858  */
859 void HFractalGui::showHelp() {
860     // Open the help page in the repository, cross-platform
861     #ifdef _WIN32
862         system("explorer
https://github.com/JkyProgrammer/HyperFractal/blob/main/README.md#help--
instructions");
863     #else
864         system("open
https://github.com/JkyProgrammer/HyperFractal/blob/main/README.md#help--
```

```
instructions");
865     #endif
866 }
867
868 /**
869  * @brief Handle the user pressing a GUI button
870  *
871  * @return True if a button press was handled, otherwise false
872  */
873 bool HFractalGui::handleButtonPresses() {
874     if (is_rendering) return false;
875     // Branch to different handling modes depending on the dialog state,
allowing certain sets of buttons to be disabled when dialogs are open
876     if (modal_view_state == MODAL_VIEW_STATE::MVS_TEXT_DIALOG) {
877         if (button_states[BUTTON_ID::BUTTON_ID_TEXT_DIALOG_CLOSE]) {
closeTextDialog(); return true; }
878     } else if (modal_view_state == MODAL_VIEW_STATE::MVS_NORMAL) {
879         if (button_states[BUTTON_ID::BUTTON_ID_RENDER]) {
startFullRender(); return true; }
880         if (button_states[BUTTON_ID::BUTTON_ID_ZOOM_IN]) { zoomIn(); return
true; }
881         if (button_states[BUTTON_ID::BUTTON_ID_ZOOM_OUT]) { zoomOut();
return true; }
882         if (button_states[BUTTON_ID::BUTTON_ID_SAVE_IMAGE]) { saveImage();
return true; }
883         if (button_states[BUTTON_ID::BUTTON_ID_SAVE_RSTATE]) {
showSaveStateDialog(); return true; }
884         if (button_states[BUTTON_ID::BUTTON_ID_LOAD_RSTATE]) {
showLoadStateDialog(); return true; }
885         if (button_states[BUTTON_ID::BUTTON_ID_UP]) { moveUp(); return
true; }
886         if (button_states[BUTTON_ID::BUTTON_ID_LEFT]) { moveLeft(); return
true; }
887         if (button_states[BUTTON_ID::BUTTON_ID_RIGHT]) { moveRight();
return true; }
888         if (button_states[BUTTON_ID::BUTTON_ID_DOWN]) { moveDown(); return
true; }
889         if (button_states[BUTTON_ID::BUTTON_ID_EQ_PRESETS]) {
enterEquationPresetDialog(); return true; }
890         if (button_states[BUTTON_ID::BUTTON_ID_ZOOM_RESET]) { resetZoom();
return true; }
891         if (button_states[BUTTON_ID::BUTTON_ID_TOGGLE_COORDS]) {
toggleCoords(); return true; }
892         if (button_states[BUTTON_ID::BUTTON_ID_EVAL_LIM_LESS]) {
evalLimitLess(); return true; }
893         if (button_states[BUTTON_ID::BUTTON_ID_EVAL_LIM_MORE]) {
evalLimitMore(); return true; }
```

```
894         if (button_states[BUTTON_ID::BUTTON_ID_HELP]) { showHelp(); return
    true; }
895         if (button_states[BUTTON_ID::BUTTON_ID_EQ_INPUTBOX]) {
    textbox_focus = TEXT_FOCUS_STATE::TFS_EQUATION; return true; }
896         if (button_states[BUTTON_ID::BUTTON_ID_CP_PRESETS]) {
    enterColourPalettePresetDialog(); return true; }
897     } else if (modal_view_state ==
    MODAL_VIEW_STATE::MVS_DATABASE_SAVE_DIALOG) {
898         if (button_states[BUTTON_ID::BUTTON_ID_SAVE_NAME_INPUTBOX]) {
    textbox_focus = TEXT_FOCUS_STATE::TFS_SAVE_NAME; return true; }
899         if (button_states[BUTTON_ID::BUTTON_ID_SAVE]) {
    saveStateToDatabase(); return true; }
900         if (button_states[BUTTON_ID::BUTTON_ID_DATABASE_CANCEL]) {
    closeDatabaseDialog(); return true; }
901     } else if (modal_view_state ==
    MODAL_VIEW_STATE::MVS_DATABASE_LOAD_DIALOG) {
902         if (button_states[BUTTON_ID::BUTTON_ID_LOAD]) {
    loadStateFromDatabase(); return true; }
903         if (button_states[BUTTON_ID::BUTTON_ID_SCROLL_DOWN]) {
    databaseLoadScrollDown(); return true; }
904         if (button_states[BUTTON_ID::BUTTON_ID_SCROLL_UP]) {
    databaseLoadScrollUp(); return true; }
905         if (button_states[BUTTON_ID::BUTTON_ID_DATABASE_CANCEL]) {
    closeDatabaseDialog(); return true; }
906     }
907
908     return false;
909 }
910
911 /**
912  * @brief Clear the contents of the button states array to prevent
    unhandled button presses hanging over to the next update
913  *
914  */
915 void HFractalGui::clearButtonStates() {
916     for (int i = 0; i < BUTTON_NUM_TOTAL; i++) {
917         button_states[i] = false;
918     }
919 }
920
921 /**
922  * @brief Unload the image buffer to prevent memory leaks
923  *
924  */
925 void HFractalGui::tryUnloadImage() {
926     UnloadImage (buffer_image);
```

```
927 }
928
929 /**
930  * @brief Unload the texture buffer to prevent memory leaks
931  *
932  */
933 void HFractalGui::tryUnloadTexture() {
934     UnloadTexture (buffer_texture);
935 }
936
937 /**
938  * @brief Handle when the user presses a key
939  *
940  * @return True if a key press was handled, false otherwise
941  */
942 bool HFractalGui::handleKeyPresses() {
943     // Escape currently editing text box when escape is pressed
944     if (IsKeyDown(KEY_ESCAPE)) { textbox_focus =
TEXT_FOCUS_STATE::TFS_NONE; return true; }
945
946     // Handle keys depending on which text box is focussed (if none, use
them for navigation)
947     if (textbox_focus == TEXT_FOCUS_STATE::TFS_NONE) {
948         for (auto key : key_map) {
949             if (IsKeyDown (key.first)) {
950                 button_states[key.second] = true;
951                 return true;
952             }
953         }
954     } else if (textbox_focus == TEXT_FOCUS_STATE::TFS_EQUATION) {
955         if (IsKeyDown(KEY_ENTER)) {
button_states[BUTTON_ID::BUTTON_ID_RENDER] = true; return true; }
956         int key = GetCharPressed();
957         if ((((int)'a' <= key && key <= (int)'c') || ((int)'x' <= key &&
key <= (int)'z') || key == 122 || (key >= 48 && key <= 57) || key == 94 ||
(key >= 40 && key <= 43) || key == 45 || key == 46 || key == 47 || key ==
'i') && !is_rendering) {
958             equation_buffer += (char)key;
959             hm->setEquation (equation_buffer);
960             lowres_hm->setEquation (equation_buffer);
961             if (!hm->isValidEquation()) console_text = "Invalid equation
input";
962             else parametersWereModified();
963         } else if (GetKeyPressed () == KEY_BACKSPACE && !is_rendering &&
```

```
equation_buffer.length() > 0) {
            equation_buffer.pop_back();
            hm->setEquation(equation_buffer);
            lowres_hm->setEquation(equation_buffer);
            if (!hm->isValidEquation()) console_text = "Invalid equation
input";
            else parametersWereModified();
        }
    } else if (textbox_focus == TEXT_FOCUS_STATE::TFS_SAVE_NAME) {
        int key = GetCharPressed();
        if (((int)'a' <= key && key <= (int)'z') || ((int)'A' <= key && key
<= (int)'Z')) {
            save_name_buffer += (char)key;
        } else if (GetKeyPressed() == KEY_BACKSPACE) {
            if (save_name_buffer.length() > 0) save_name_buffer.pop_back();
        }
    }
    return false;
}

/**
 * @brief Start the GUI and run the mainloop.
 * Blocks on current thread
 *
 * @return Integer showing exit status
 */
int HFractalGui::guiMain(char* path) {
    // Run the setup code
    configureGUI(path);
    parametersWereModified();
    while(!WindowShouldClose()) { // Loop until the application closes
        checkWindowResize();
        if (!is_rendering && modal_view_state == MVS_NORMAL) {
            bool click_handled = handleClickNavigation();
            // Defocus the textbox if a click is handled somewhere
            if (click_handled) { textbox_focus =
TEXT_FOCUS_STATE::TFS_NONE; }
        }
        handleKeyPresses();
        bool button_pressed = handleButtonPresses();
        // Defocus the textbox if a button press is handled
        if (button_pressed &&
!button_states[BUTTON_ID::BUTTON_ID_EQ_INPUTBOX] &&
```

```
!button_states[BUTTON_ID::BUTTON_ID_SAVE_NAME_INPUTBOX]) { textbox_focus =
TEXT_FOCUS_STATE::TFS_NONE; }
1002            clearButtonStates();
1003            // If a render is in progress, update the status of it
1004            if (is_rendering) updateFullRender();
1005            // Finally, draw everything
1006            drawInterface();
1007        }
1008
1009        // Release resources and close
1010        tryUnloadImage();
1011        tryUnloadTexture();
1012        CloseWindow();
1013        return 0;
1014    }
1015
1016    /**
1017     * @brief Construct a new GUI object
1018     *
1019     */
1020    HFractalGui::HFractalGui() {}
1021
1022    /**
1023     * @brief Method to start the GUI, isolates the GUI module from the main
module to prevent linker conflicts with raylib
1024     *
1025     * @return Integer showing exit status
1026     */
1027    int guiMain (char* path) {
1028        HFractalGui gui = HFractalGui ();
1029        int res = gui.guiMain(path);
1030        return res;
1031    }
```

```cpp
// src/gui.hh

#ifndef GUI_H
#define GUI_H

#include <map>

#define RAYGUI_IMPLEMENTATION
#define RAYGUI_SUPPORT_ICONS
#include "../lib/raygui.h"
#include "../lib/ricons.h"

#include "hyperfractal.hh"
#include "utils.hh"
#include "database.hh"

#define SCALE_STEP_FACTOR 1.5       // Factor by which scaling changes
#define SCALE_DEPTH_LIMIT 1.0e15    // Limit to prevent user from going too deep due to limited precision
#define MOVE_STEP_FACTOR 0.1        // Factor by which position changes
#define WINDOW_INIT_WIDTH 900       // Initial window - width
#define WINDOW_INIT_HEIGHT 550      //                - height
#define BUTTON_HEIGHT 30            // Height of a single button in the interface
#define ELEMENT_NUM_VERTICAL 15     // Number of vertical elements
#define BUTTON_NUM_TOTAL 25         // Total number of buttons in the interface
#define CONTROL_MIN_WIDTH 400       // Minimum width of the control panel
#define CONTROL_MIN_HEIGHT BUTTON_HEIGHT*ELEMENT_NUM_VERTICAL // Minimum height of the panel
#define DIALOG_TEXT_SIZE 25         // Size of text in dialog windows

// Enum listing button IDs to abstract and make code clearer
enum BUTTON_ID {
    BUTTON_ID_RENDER = 0,
    BUTTON_ID_ZOOM_IN,
    BUTTON_ID_ZOOM_OUT,
    BUTTON_ID_SAVE_IMAGE,
    BUTTON_ID_SAVE_RSTATE,
    BUTTON_ID_LOAD_RSTATE,
    BUTTON_ID_UP,
    BUTTON_ID_LEFT,
    BUTTON_ID_RIGHT,
```

```
    BUTTON_ID_DOWN,
    BUTTON_ID_EQ_PRESETS,
    BUTTON_ID_ZOOM_RESET,
    BUTTON_ID_TOGGLE_COORDS,
    BUTTON_ID_EVAL_LIM_LESS,
    BUTTON_ID_EVAL_LIM_MORE,
    BUTTON_ID_HELP,
    BUTTON_ID_TEXT_DIALOG_CLOSE,
    BUTTON_ID_EQ_INPUTBOX,
    BUTTON_ID_CP_PRESETS,
    BUTTON_ID_SAVE_NAME_INPUTBOX,
    BUTTON_ID_SAVE,
    BUTTON_ID_LOAD,
    BUTTON_ID_SCROLL_DOWN,
    BUTTON_ID_SCROLL_UP,
    BUTTON_ID_DATABASE_CANCEL
};

// Enum listing GUI states for cases when a dialog or modal is open (i.e. to disable certain interface elements)
enum MODAL_VIEW_STATE {
    MVS_NORMAL,
    MVS_TEXT_DIALOG,
    MVS_DATABASE_SAVE_DIALOG,
    MVS_DATABASE_LOAD_DIALOG,
    MVS_EQUATION_PRESET_SELECTOR,
    MVS_COLOUR_PRESET_SELECTOR
};

// Enum listing text focus states to enable/disable input to specific fields
enum TEXT_FOCUS_STATE {
    TFS_NONE,
    TFS_EQUATION,
    TFS_SAVE_NAME
};

// Class managing the GUI environment
class HFractalGui {
private:
    // Lists which keys on the keyboard map to which interface buttons
    std::map<KeyboardKey, BUTTON_ID> key_map = {
        {KEY_ENTER, BUTTON_ID::BUTTON_ID_RENDER},
```

```
81          {KEY_EQUAL, BUTTON_ID::BUTTON_ID_ZOOM_IN},
82          {KEY_MINUS, BUTTON_ID::BUTTON_ID_ZOOM_OUT},
83          {KEY_UP, BUTTON_ID::BUTTON_ID_UP},
84          {KEY_DOWN, BUTTON_ID::BUTTON_ID_DOWN},
85          {KEY_LEFT, BUTTON_ID::BUTTON_ID_LEFT},
86          {KEY_RIGHT, BUTTON_ID::BUTTON_ID_RIGHT},
87          {KEY_LEFT_BRACKET, BUTTON_ID::BUTTON_ID_EVAL_LIM_LESS},
88          {KEY_RIGHT_BRACKET, BUTTON_ID::BUTTON_ID_EVAL_LIM_MORE}
89      };
90
91      HFractalMain* hm; // Pointer to main rendering environment
92      HFractalMain* lowres_hm; // Pointer to an identical rendering
    environment, but with a lower resolution for preview renders
93
94      std::string dialog_text; // Text to show in the text dialog widget
95      std::string console_text; // Text to show in the application console
96      std::string equation_buffer; // Contains the equation being used by both
    renderer classes
97      std::string save_name_buffer; // Contains the text shown/edited in the
    name field in the save render state dialog
98      bool button_states[BUTTON_NUM_TOTAL]; // Contains the current states of
    every button in the GUI (true for pressed, false for not pressed)
99      Image buffer_image; // Image being used by raygui for displaying the
    render result
100     Texture2D buffer_texture; // Texture being used by raygui for displaying
    the render result
101     bool is_rendering; // Stores whether the GUI is currently waiting on a
    full-resolution render (and thus should freeze controls)
102     bool is_outdated_render; // Stores whether the GUI is showing a preview
    render (i.e. needs a full-resolution render to be run by the user)
103     TEXT_FOCUS_STATE textbox_focus; // Stores the currently focussed text
    box
104     int render_percentage; // Stores the percentage completion of the
    current render
105     bool showing_coordinates; // Stores whether coordinates are currently
    being shown on the mouse cursor
106     MODAL_VIEW_STATE modal_view_state; // Stores the current modal state of
    the GUI, allowing certain controls to be enabled and disabled in different
    modes
107     int image_dimension; // Stores the size of the image, used for sizing
    the window, scaling and rendering images, and positioning elements
108     int control_panel_width; // Stores the width of the control panel
109     CP_PRESETS selected_palette; // Determines the colour palette in which
    the GUI is currently displaying the rendered image
110     HFractalDatabase database; // Database which manages saved profile
    states
111     long selected_profile_id; // Records the ID of the profile currently
```

```
    selected in the load render state dialog
112     int database_load_dialog_scroll; // Records the current amount of scroll
    in the load render state dialog
113 
114     void configureStyling(); // Configures the GUI styling from a stylesheet
    provided by raylib's creator as part of the library
115     void configureGUI(char*); // Configures the GUI and initialises all
    class variables ready for the first GUI mainloop update
116 
117     void parametersWereModified(); // Marks the GUI as using an outdated
    render and triggers a preview render update
118     bool updatePreviewRender(); // Rerenders the preview image
119     bool startFullRender(); // Triggers a full resolution render
120     bool updateFullRender(); // Updates the image and texture buffers from
    the partially-finished rendering environment image, and finalises if the
    render has completed
121     void reloadImageFrom(HFractalMain*); // Automatically fetch and reload
    the image and texture buffers from a given rendering environment
122 
123     void checkWindowResize(); // Check to see if the window has been
    resized, and handle it
124 
125     bool handleClickNavigation(); // Check to see if the user has clicked
    somewhere on the image, and jump to focus that location if so
126     bool handleButtonPresses(); // Handle any interface button presses the
    user has made since the last update
127     bool handleKeyPresses(); // Handle any keyboard key presses the user has
    made since the last update
128     void drawInterface(); // Draw the entire interface, called each update
129 
130     Image getImage(HFractalMain*); // Extract image data from a rendering
    environment
131 
132     void enterEquationPresetDialog(); // Show the equation preset selector
    and disable other GUI controls
133     void escapeEquationPresetDialog(int); // Close the equation preset
    selector and return to normal GUI mode
134     void enterColourPalettePresetDialog(); // Show the colour palette preset
    selector and disable other GUI controls
135     void escapeColourPalettePresetDialog(int); // Close the colour palette
    preset selector and return to normal GUI mode
136     void launchTextDialog(std::string); // Show a text dialog over the
    window with a given string as text
137     void closeTextDialog(); // Close the text dialog currently being shown
138 
139     void zoomIn(); // Handler for Zoom In button
140     void zoomOut(); // Handler for Zoom Out button
```

```
141     void resetZoom(); // Handler for Reset Zoom button
142     void saveImage(); // Handler for Save Image button
143 
144     void moveUp(); // Handler for Move Up button
145     void moveLeft(); // Handler for Move Left button
146     void moveRight(); // Handler for Move Right button
147     void moveDown(); // Handler for Move Down button
148 
149     void toggleCoords(); // Handler for Show/Hide Coordinates button
150 
151     void evalLimitLess(); // Handler for '<' button
152     void evalLimitMore(); // Handler for '>' button
153 
154     void showHelp(); // Handler for Help & Instructions button
155     void clearButtonStates(); // Clears current button states to ignore
    unhandled button presses
156 
157     void tryUnloadImage(); // Unload the image buffer, prevents memory leaks
158     void tryUnloadTexture(); // Unload the texture buffer, prevents memory
    leaks
159 
160     void showSaveStateDialog(); // Make the save render state dialog visible
161     void showLoadStateDialog(); // Make the load render state dialog visible
162     void saveStateToDatabase(); // Save the current render state to the
    database
163     void loadStateFromDatabase(); // Load the selected config profile from
    the database to be the current render state
164     void closeDatabaseDialog(); // Hide the save/load render state dialog
165     void databaseLoadScrollDown(); // Scroll down in the load render state
    dialog
166     void databaseLoadScrollUp(); // Scroll up in the load render state
    dialog
167 public:
168     int guiMain(char*); // Start and run the entire GUI. Blocks on current
    thread
169 
170     HFractalGui(); // Basic constructor
171 };
172 
173 #endif
```

```
// src/guimain.hh

#ifndef GUIMAIN_H
#define GUIMAIN_H

// GUI Main function to isolate the GUI module from the main module to prevent
linker conflicts
int guiMain(char*);

#endif
```

```cpp
// src/hyperfractal.cc

#include "hyperfractal.hh"

#include <iostream>
#include <iomanip>
#include <chrono>
#include <thread>

#include "utils.hh"

using namespace std;
using namespace std::chrono;

/**
 * @brief Main function called when each worker thread starts. Contains code to actually fetch and render pixels
 *
 */
void HFractalMain::threadMain () {
    // Pre-compute constants to increase performance
    long double p = 2/(zoom*resolution);
    long double q = (1/zoom)-offset_x;
    long double r = (1/zoom)+offset_y;

    // Get the next unrendered pixel
    int next = img->getUncompleted();
    while (next != -1) {
        // Find the x and y coordinates based on the pixel index
        int x = next%resolution;
        int y = next/resolution;
        // Apply the mathematical transformation of offsets and zoom to find a and b, which form a coordinate pair representing this pixel in the complex plane
        long double a = (p*x) - q;
        long double b = r - (p*y);
        // Construct the initial coordinate value, and perform the evaluation on the main equation
        complex<long double> c = complex<long double> (a,b);
        int res = (main_equation->evaluate (c, eval_limit));
        // Set the result back into the image class, and get the next available unrendered pixel
        img->set (x, y, res);
```

```
39          next = img->getUncompleted();
40      }
41
42      // When there appear to be no more pixels to compute, mark this thread
as completed
43      thread_completion[std::this_thread::get_id()] = true;
44
45      // Check to see if any other threads are still rendering, if not then
set the flag to mark the environment as no longer rendering
46      bool is_incomplete = false;
47      for (auto p : thread_completion) is_incomplete |= !p.second;
48      if (!is_incomplete) is_rendering = false;
49  }
50
51  /**
52   * @brief Generate a fractal image based on all the environment parameters
53   *
54   * @param wait Whether to wait and block the current thread until the image
has been fully computed, useful if you want to avoid concurrency somewhere
else (functionality hiding)
55   * @return Integer representing status code, 0 for success, else for failure
56   */
57  int HFractalMain::generateImage (bool wait=true) {
58      if (getIsRendering()) { std::cout << "Aborting!" << std::endl; return 2;
} // Prevent overlapping renders from starting
59      // Output a summary of the rendering parameters
60      std::setprecision (100);
61      std::cout << "Rendering with parameters: " << std::endl;
62      std::cout << "Resolution=" << resolution << std::endl;
63      std::cout << "EvaluationLimit=" << eval_limit << std::endl;
64      std::cout << "Threads=" << worker_threads << std::endl;
65      std::cout << "Zoom="; printf ("%Le", zoom); std::cout << std::endl;
66      std::cout << "OffsetX="; printf ("%.70Lf", offset_x); std::cout <<
std::endl;
67      std::cout << "OffsetY="; printf ("%.70Lf", offset_y); std::cout <<
std::endl;
68
69      // Abort rendering if the equation is invalid
70      if (!isValidEquation()) { std::cout << "Aborting!" << std::endl; return
1; }
71
72      // Mark the environment as now rendering, locking resources/parameters
73      is_rendering = true;
74
```

```
75      // Clear and reinitialise the image class with the requested resolution
76      if (img != NULL) img->~HFractalImage();
77      img = new HFractalImage (resolution, resolution);
78
79      // Clear the thread pool, and populate it with fresh worker threads
80      thread_pool.clear();
81      thread_completion.clear();
82      for (int i = 0; i < worker_threads; i++) {
83          std::thread *t = new std::thread(&HFractalMain::threadMain, this);
84          thread_completion[t->get_id()] = false;
85          thread_pool.push_back(t);
86      }
87
88      // Optionally, wait for the render to complete before returning
89      if (wait) {
90          while (true) {
91              // If enabled at compile time, show a progress bar in the
terminal
92              #ifdef TERMINAL_UPDATES
93              float percent = getImageCompletionPercentage();
94              std::cout << "\r";
95              std::cout << "Working: ";
96              for (int k = 2; k <= 100; k+=2) { if (k <= percent) std::cout <<
"▉"; else std::cout << "_"; }
97              std::cout << " | ";
98              std::cout << round(percent) << "%";
99              #endif
100             // Break out when the image has been fully completed (all pixels
computed)
101             if (img->isDone()) break;
102             crossPlatformDelay (10);
103         }
104         // Wait for all the threads to join, then finish up
105         for (auto th : thread_pool) th->join();
106         is_rendering = false;
107     }
108     std::cout << std::endl << "Rendering done." << std::endl;
109     return 0;
110 }
111
112 /**
113  * @brief Construct a new rendering environment, with blank parameters
114  *
```

```
 */
HFractalMain::HFractalMain () {
    resolution = 1;
    offset_x = 0;
    offset_y = 0;
    zoom = 1;
    img = NULL;
}

/**
 * @brief Convert the raw data stored in the image class into a coloured RGBA 32 bit image using a particular colour scheme preset
 *
 * @param colour_preset The colour scheme to use
 * @return uint32_t* Pointer to the image stored in memory as an array of 4-byte chunks
 */
uint32_t* HFractalMain::getRGBAImage (int colour_preset) {
    // Return a blank result if the image is uninitialised, other conditions should ensure this never occurs
    if (img == NULL) {
        return (uint32_t *)malloc(0);
    }

    // Copy parameters to local
    int size = resolution;
    int limit = eval_limit;

    // Construct a pixel buffer with RGBA channels
    uint32_t *pixels = (uint32_t *)malloc(size*size*sizeof(uint32_t));
    for (int x = 0; x < size; x++) {
        for (int y = 0; y < size; y++) {
            int v = img->get(x,y);
            pixels[(y*size)+x] = (v == limit) ? 0x000000ff : HFractalImage::colourFromValue(v, colour_preset);

            // If the pixel has not been computed, make it transparent
            if (img->completed[(y*size)+x] != 2) pixels[(y*size)+x] = 0;
        }
    }

    // Return the pointer to the pixel buffer
    return pixels;
```

```
154 }
155
156 /**
157  * @brief Get the percentage of pixels in the image which have been computed
158  *
159  * @return Unrounded percentage
160  */
161 float HFractalMain::getImageCompletionPercentage () {
162     if (img == NULL) return 100;
163     return ((float)(img->getInd())/(float)(resolution*resolution))*100;
164 }
165
166 /**
167  * @brief Automatically write an image to a generated file address.
168  * File path will be dynamically constructed so that the file name is
    unique, timestamped, and placed on the user's desktop
169  *
170  * @param type Image format to write image out to
171  * @return True for success, false for failure
172  */
173 bool HFractalMain::autoWriteImage (IMAGE_TYPE type) {
174     string image_name = "Fractal render from ";
175
176     // Get current system time
177     auto time = system_clock::to_time_t (system_clock::now());
178     string c_time = string (ctime (&time));
179
180     // Separate ctime result into components
181     vector<string> time_components;
182     string current_component = "";
183     for (char c : c_time) {
184         if (c == ' ') {
185             time_components.push_back (current_component);
186             current_component = "";
187         } else if (c != '\n') current_component.push_back (c != ':' ? c :
    '.');
188     }
189     time_components.push_back (current_component);
190
191     // Get milliseconds, not part of ctime
192     system_clock::duration dur = system_clock::now().time_since_epoch();
193     seconds s = duration_cast<seconds> (dur);
```

```
194     dur -= s;
195     milliseconds ms = duration_cast<milliseconds> (dur);
196 
197     // Components are in the form: dayofweek month day hour:minute:second
    year
198     image_name += time_components[2] + " ";
199     image_name += time_components[1] + " ";
200     image_name += time_components[4] + " ";
201     image_name += "at ";
202     image_name += time_components[3];
203     image_name += ".";
204     image_name += to_string(ms.count());
205 
206     cout << image_name << endl;
207 
208     string image_path = "";
209 
210     image_path += getDesktopPath();
211     image_path += image_name;
212 
213     // Call into the image's writer to write out data
214     switch (type) {
215     case PGM:
216         image_path += ".pgm";
217         return img->writePGM (image_path);
218     default:
219         return false;
220     }
221 }
```

```cpp
// src/hyperfractal.hh

#ifndef HYPERFRACTAL_H
#define HYPERFRACTAL_H

#include <string>
#include <thread>
#include <vector>
#include <map>

#include "image.hh"
#include "fractal.hh"
#include "utils.hh"
#include "equationparser.hh"

// When defined, progress updates will be written to terminal.
#define TERMINAL_UPDATES

// Class defining a fractal rendering environment, fully encapsulated
class HFractalMain {
private:
    int resolution; // Horizontal and vertical dimension of the desired image
    long double offset_x; // Horizontal offset in the complex plane
    long double offset_y; // Vertical offset in the complex plane
    long double zoom; // Scaling value for the image (i.e. zooming in)

    std::string eq; // String equation being used
    HFractalEquation *main_equation; // Actual pointer to the equation manager class being used for computation

    int worker_threads; // Number of worker threads to be used for computation
    int eval_limit; // Evaluation limit for the rendering environment

    HFractalImage *img = new HFractalImage(0,0); // Pointer to the image class containing data for the rendered image

    std::vector<std::thread*> thread_pool; // Thread pool containing currently active threads
    std::map<std::thread::id, bool> thread_completion; // Map of which threads have finished computing pixels
    bool is_rendering = false; // Marks whether there is currently a render ongoing (locking resources to prevent concurrent modification e.g. changing resolution mid-render)
```

```
38
39      void threadMain (); // Method called on each thread when it starts,
contains the worker/rendering code
40
41  public:
42      int generateImage (bool); // Perform the render, and optionally block the
current thread until it is done
43
44      HFractalMain (); // Base initialiser
45
46      int getResolution () { return resolution; } // Inline methods to get/set
the resolution
47      void setResolution (int resolution_) { if (!getIsRendering()) resolution
= resolution_; }
48
49      long double getOffsetX () { return offset_x; } // Inline methods to
get/set the x offset
50      void setOffsetX (long double offset_x_) { if (!getIsRendering()) offset_x
= offset_x_; }
51
52      long double getOffsetY () { return offset_y; } // Inline methods to
get/set the y offset
53      void setOffsetY (long double offset_y_) { if (!getIsRendering()) offset_y
= offset_y_; }
54
55      long double getZoom () { return zoom; } // Inline methods to get/set the
zoom
56      void setZoom (long double zoom_) { if (!getIsRendering()) zoom = zoom_; }
57
58      std::string getEquation () { return eq; } // Inline methods to get/set
the equation
59      void setEquation (std::string eq_) {
60          if (!getIsRendering()) {
61              eq = eq_;
62              main_equation = HFractalEquationParser::extractEquation (eq);
63              if (main_equation == NULL) return;
64              // Detect if the equation matches the blueprint of a preset
65              int preset = -1;
66              for (int i = 0; i < NUM_EQUATION_PRESETS; i++) {
67                  if (eq == equationPreset ((EQ_PRESETS)i, false)) {
68                      preset = i;
69                      break;
70              main_equation->setPreset (preset);
71          }
72      }
```

```
73      main_equation->setPreset (preset);
74          }
75      }
76
77      int getWorkerThreads () { return worker_threads; } // Inline methods to
get/set the number of worker threads
78      void setWorkerThreads (int wt_) { if (!getIsRendering()) worker_threads =
wt_; }
79
80      int getEvalLimit () { return eval_limit; } // Inline methods to get/set
the evaluation limit
81      void setEvalLimit (int el_) { if (!getIsRendering()) eval_limit = el_; }
82
83      bool isValidEquation () { return main_equation != NULL; } // Check if the
equation the user entered was parsed correctly last time it was set
84
85      bool getIsRendering() { return is_rendering; } // Get if there is
currently a render happening in this environment
86
87      uint32_t* getRGBAImage (int); // Return a pointer to a 32 bit RGBA
formatted image, produced using a particular colour scheme preset, from the
generated image
88
89      float getImageCompletionPercentage (); // Get the current percentage of
pixels that have been actually computed
90
91      bool autoWriteImage (IMAGE_TYPE); // Automatically write out the render
to desktop using a particular image type
92  };
93  #endif
```

```cpp
// src/image.cc

#include "image.hh"

#include <ostream>
#include <math.h>

/**
 * @brief Set the value of a pixel, and automatically mark it as complete
 *
 * @param x Horizontal coordinate
 * @param y Vertical coordinate
 * @param p Value of the pixel to assign
 */
void HFractalImage::set(int x, int y, uint16_t p) {
    int offset = ((y*width)+x);
    data_image[offset] = p;
    completed[offset] = 2;
}

/**
 * @brief Get the value of the pixel at the specified coordinates, as measured from top-left
 *
 * @param x Horizontal coordinate
 * @param y Vertical coordinate
 * @return The value of the pixel at the coordinates
 */
uint16_t HFractalImage::get(int x, int y) {
    return data_image[(y*width)+x];
}

/**
 * @brief Initialise a new image with a specified width and height
 *
 * @param w Horizontal size
 * @param h Vertical size
 */
HFractalImage::HFractalImage(int w, int h) {
    width = w;
    height = h;
    c_ind = 0;
```

```
42      data_image = new uint16_t[width*height];
43      completed = new uint8_t[width*height];
44      // Clear both buffers
45      for (int i = 0; i < width*height; i++) { data_image[i] = 0xffff;
completed[i] = 0; }
46  }
47
48  /**
49   * @brief Write the contents of the image buffer out to a PGM file (a
minimal image format using grayscale linear colour space)
50   *
51   * @param path Path to the output file
52   * @return True for success, false for failure
53   */
54  bool HFractalImage::writePGM (std::string path) {
55      // Abort if the image is incomplete
56      if (!isDone()) return false;
57      FILE *img_file;
58      img_file = fopen(path.c_str(),"wb");
59
60      // Write the header
61      fprintf(img_file,"P5\n");
62      fprintf(img_file,"%d %d\n",width,height);
63      fprintf(img_file,"65535\n");
64
65      // Write each pixel
66      for(int y = 0; y < height; y++){
67          for(int x = 0; x < width; x++){
68              uint16_t p = data_image[(y*width)+x];
69
70              fputc (p & 0x00ff, img_file);
71              fputc ((p & 0xff00) >> 8, img_file);
72          }
73      }
74
75      // Close and return success
76      fclose(img_file);
77      return true;
78  }
79
80  /**
81   * @brief Create an RGBA 32 bit colour from hue, saturation, value
components
```

```
 *
 * @param h Hue value
 * @param s Saturation value
 * @param v Value (brightness) value
 * @return A 32 bit colour in RGBA form
 */
uint32_t HFractalImage::HSVToRGB (float h, float s, float v) { // TODO: Test this
    float c = v * s;
    float h_ = fmod(h,1)*6;
    float x = c * (1 - fabsf(fmodf(h_, 2)-1));
    float m = v - c;

    float r;
    float g;
    float b;

    if (h_ >= 0 && h_ < 1) {
        r = c; g = x; b = 0;
    } else if (h_ < 2) {
        r = x; g = c; b = 0;
    } else if (h_ < 3) {
        r = 0; g = c; b = x;
    } else if (h_ < 4) {
        r = 0; g = x; b = c;
    } else if (h_ < 5) {
        r = x; g = 0; b = c;
    } else if (h_ < 6) {
        r = c; g = 0; b = x;
    }

    r = r + m;
    g = g + m;
    b = b + m;

    uint32_t final_value = 0x000000ff;
    final_value |= (int)(r*255) << (8*3);
    final_value |= (int)(g*255) << (8*2);
    final_value |= (int)(b*255) << (8*1);

    return final_value;
}
```

```
123
124 /**
125  * @brief Convert a computed value (i.e. from the image_data buffer) into a
    renderable RGBA 32 bit colour
126  *
127  * @param value The value to convert
128  * @param colour_preset Colour palette preset to map colour onto
129  * @return The converted colour as a 32 bit integer
130  */
131 uint32_t HFractalImage::colourFromValue (uint16_t value, int colour_preset)
    {
132     uint32_t col = 0x000000ff;
133     if (colour_preset == 0) {
134         col |= 0x3311ff00;
135         col |= ((value % 256) << (8*3)) + 0x33000000;
136     } else if (colour_preset == 1) {
137         uint8_t looped = 255-(uint8_t)(value % 256);
138         col |= looped << (8*3);
139         col |= (2*looped) << (8*2);
140         col |= 0x00007000;
141     } else if (colour_preset == 2) {
142         float hue = (float)value/(float)0xffff;
143         hue = fmod(512*hue, 1);
144         col = HFractalImage::HSVToRGB (hue, 0.45, 0.8);
145     } else if (colour_preset == 3) {
146         uint8_t looped = 255-(uint8_t)(value % 256);
147         col |= looped << (8*1); // B
148         col |= looped << (8*2); // G
149         col |= looped << (8*3); // R
150     } else if (colour_preset == 4) {
151         uint8_t looped = (uint8_t)(value % 256);
152         col |= looped << (8*1); // B
153         col |= looped << (8*2); // G
154         col |= looped << (8*3); // R
155     }
156
157
158     return col;
159 }
160
161 /**
162  * @brief Destroy the image class, freeing the buffers
```

```
 */
HFractalImage::~HFractalImage () {
    free (data_image);
    free (completed);
}

/**
 * @brief Fetch the index (i.e. (y*width)+x) of the next pixel which needs to be computed
 *
 * @return The index of the next pixel to compute, -1 if there is no available pixel
 */
int HFractalImage::getUncompleted () {
    // Lock resources to prevent collisions
    mut.lock();
    int i = -1;
    // Find the next available pixel index and increment c_ind
    if (c_ind < height*width) {
        i = c_ind;
        c_ind++;
    }
    // Unlock before returning
    mut.unlock();
    return i;
}

/**
 * @brief Check every pixel to see if the image is fully computed. Use with caution, especially with large images
 *
 * @return True if the image is complete, false otherwise
 */
bool HFractalImage::isDone () {
    // Iterate over every pixel to check its status
    for (int i = 0; i < height*width; i++) {
        if (completed[i] != 2) {
            // Fail if the pixel is not complete
            return false;
        }
    }
    // Succeed if every pixel is fully computed
    return true;
```

```
}

/**
 * @brief Get the current completion index of the image
 *
 * @return The current completion index
 */
int HFractalImage::getInd () { return c_ind; }
```

```
1  // src/image.hh
2
3  #ifndef IMAGE_H
4  #define IMAGE_H
5
6  #include <mutex>
7
8  // Class containing information about an image currently being generated
9  class HFractalImage {
10 private:
11     int width; // Width of the image
12     int height; // Heigh of the image
13     uint16_t * data_image; // Computed data values of the image
14     int c_ind = 0; // Index of the next pixel to be sent out to a rendering
   thread
15     std::mutex mut; // Mutex object used to lock class resources during
   multi-threading events
16
17 public:
18     HFractalImage (int, int); // Constructor, creates a new image buffer of
   the specified size
19     ~HFractalImage (); // Destructor, destroys and deallocates resources used
   in the current image
20
21     void set (int, int, uint16_t); // Set the value of a pixel
22     uint16_t get (int, int); // Get the value of a pixel
23     uint8_t * completed; // Stores the completion status of each pixel, 0 =
   not computed, 1 = in progress, 2 = computed
24     int getUncompleted (); // Get the index of an uncomputed pixel, to be
   sent to a rendering thread, and update completion data
25     bool isDone (); // Check if the image has been completed or not
26     int getInd (); // Get the current completion index
27     bool writePGM (std::string); // Write out the contents of the data buffer
   to a simple image file, PGM format, with the given path
28
29     static uint32_t HSVToRGB (float h, float s, float v); // Create a 32 bit
   RGB colour from hue, saturation, value components
30     static uint32_t colourFromValue (uint16_t, int); // Convert a computed
   value into a 32 bit RGBA colour value, using the specified palette
31 };
32
33 #endif
```

```cpp
// src/main.cc

#include <iostream>

#include "hyperfractal.hh"
#include "guimain.hh"
#include "utils.hh"

using namespace std;

/**
 * Naming Convention:
 * Classes & Structs - CapitalisedCamelCase
 * Variables - snake_case
 * Functions - uncapitalisedCamelCase
 * Constants - SCREAMING_SNAKE_CASE
 *
 **/

int main (int argc, char *argv[]) {
    if (argc == 8) {
        // If we have the required arguments, run a console-only render
        HFractalMain hm;
        int argument_error = 0;
        try {
            hm.setResolution (stoi (argv[1]));
            if (hm.getResolution() <= 0) throw runtime_error("Specified resolution too low.");
            argument_error++;
            hm.setOffsetX (stod (argv[2]));
            argument_error++;
            hm.setOffsetY (stod (argv[3]));
            argument_error++;
            hm.setZoom (stod (argv[4]));
            argument_error++;
            hm.setEquation (string (argv[5]));
            if (!hm.isValidEquation()) throw runtime_error("Specified equation is invalid.");
            argument_error++;
            hm.setWorkerThreads (stoi (argv[6]));
            if (hm.getWorkerThreads() <= 0) throw runtime_error("Must use at least one worker thread.");
```

```
            argument_error++;
            hm.setEvalLimit (stoi (argv[7]));
            if (hm.getEvalLimit() <= 0) throw runtime_error("Must use at least one evaluation iteration.");
            argument_error++;
            hm.generateImage(true);
            return !hm.autoWriteImage (IMAGE_TYPE::PGM);
        } catch (runtime_error e) {
            cout << "Parameter error on argument number " << argument_error << ":" << endl;
            cout << "   " << e.what() << endl;
            return 1;
        }
    } else if (argc != 1) {
        // If we have only some arguments, show the user what arguments they need to provide
        cout << "Provide all the correct arguments please:" << endl;
        cout << "int resolution, long double offset_x, long double offset_y, long double zoom, string equation, int worker_threads, int eval_limit" << endl;
        return 1;
    } else {
        // Otherwise, start the GUI
        cout << trimExecutableFromPath(argv[0]) << endl;
        return guiMain(trimExecutableFromPath(argv[0]));
    }
}
```

```cpp
// src/utils.cc

#include "utils.hh"

#ifdef _WIN32
#include <windows.h>
#include <shlobj.h>
#else
    #include <unistd.h>
#endif

using namespace std;

/**
 * @brief Get the details of an equation preset
 *
 * @param p The preset to return
 * @param t True - return the name of the preset, False - return the string equation of it instead
 * @return Either the name or the equation string of the equation preset
 */
string equationPreset (EQ_PRESETS p, bool t) {
    switch (p) {
    case EQ_MANDELBROT:
        return t ? "Mandelbrot" : "(z^2)+c";
    case EQ_JULIA_1:
        return t ? "Juila 1" : "(z^2)+(0.285+0.01i)";
    case EQ_JULIA_2:
        return t ? "Julia 2" : "(z^2)+(-0.70176-0.3842i)";
    case EQ_RECIPROCAL:
        return t ? "Reciprocal" : "1/((z^2)+c)";
    case EQ_ZPOWER:
        return t ? "Z Power" : "(z^z)+(c-0.5)";
    case EQ_BARS:
        return t ? "Bars" : "z^(c^2)";
    case EQ_BURNINGSHIP_MODIFIED:
        return t ? "Burning Ship Modified" : "((x^2)^0.5-((y^2)^0.5)i)^2+c";
    default:
        return "NONE";
    }
    return "";
}
```

```
42
43 /**
44  * @brief Return the name of a colour palette preset
45  *
46  * @param p Preset to return
47  * @return The string name of the colour palette
48  */
49 string colourPalettePreset (CP_PRESETS p) {
50     switch (p) {
51     case CP_VAPORWAVE:
52         return "Vaporwave";
53     case CP_YELLOWGREEN:
54         return "Yellow-Green";
55     case CP_RAINBOW:
56         return "Rainbow";
57     case CP_GREYSCALE_BRIGHT:
58         return "Greyscale Bright";
59     case CP_GREYSCALE_DARK:
60         return "Greyscale Dark";
61     default:
62         return "NONE";
63     }
64     return "";
65 }
66
67 /**
68  * @brief Wrap text given a certain line length
69  *
70  * @param s String to wrap
71  * @param line_length Numbere of characters which limit the length of the
    line
72  * @return string
73  */
74 string textWrap (string s, int line_length) {
75     string output = "";
76     int line_offset = 0;
77     for (char c : s) {
78         if (c == '\n') line_offset = -1;
79         if (line_offset == line_length-1) { if (c != ' ') output += "-";
    output += "\n"; line_offset = 0; }
80         if (!(line_offset == 0 && c == ' ')) {
81             output += c;
```

```
            line_offset++;
        }
    }
    return output;
}

/**
 * @brief Get the desktop path of the user
 *
 * @return The user's desktop path
 */
string getDesktopPath () {
    #ifdef _WIN32
    static char path[MAX_PATH+1];
    if (SHGetSpecialFolderPathA(HWND_DESKTOP, path, CSIDL_DESKTOP, FALSE))
        return string(path) + string("\\");
    else
        return "";
    #else
    return string(getenv ("HOME")) + string("/Desktop/");
    #endif
}

/**
 * @brief Delay for a number of millisecods, across any platform
 *
 * @param milliseconds Time to delay
 */
void crossPlatformDelay(int milliseconds) {
    #ifdef _WIN32
        Sleep(milliseconds);
    #else
        usleep(milliseconds * 1000);
    #endif
}

/**
 * @brief Trim the executable name from the end of the path, returning just the working directory
 *
 * @param path Input path from command line arguments
 * @return Trimmed path
```

```
 */
char* trimExecutableFromPath(char* path) {
    int str_len = 0;
    while (true) {
        if (path[str_len] == '\0') break;
        str_len++;
    }

    uint32_t slash_index = -1;
    for (int i = str_len-1; i >= 0; i--) {
        if (path[i] == '\\' || path[i] == '/') {
            slash_index = i;
            break;
        }
    }

    char* result = (char*)malloc(slash_index+2);

    for (int i = 0; i < slash_index+1; i++) {
        result[i] = path[i];
    }
    result[slash_index+1] = '\0';

    return result;
}
```

```
// src/utils.hh

#ifndef UTILS_H
#define UTILS_H

#include <string>

#define NUM_EQUATION_PRESETS 7
#define NUM_COLOUR_PRESETS 5

// Wrap text along a line length
std::string textWrap (std::string, int);

// Enum describing possible equation presets
enum EQ_PRESETS {
    EQ_MANDELBROT = 0, // "(z^2)+c"
    EQ_JULIA_1, // "(z^2)+(0.285+0.01i)"
    EQ_JULIA_2, // "(z^2)+(-0.70176-0.3842i)"
    EQ_RECIPROCAL, // "1/((z^2)+c)"
    EQ_ZPOWER, // "(z^z)+(c-0.5)"
    EQ_BARS, // "z^(c^2)"
    EQ_BURNINGSHIP_MODIFIED // "((x^2)^0.5-((y^2)^0.5)i)^2+c"
};

// Enum describing possible colour palette presets
enum CP_PRESETS {
    CP_VAPORWAVE = 0,
    CP_YELLOWGREEN,
    CP_RAINBOW,
    CP_GREYSCALE_BRIGHT,
    CP_GREYSCALE_DARK
};

// Enum describing available image types which can be saved to disk
enum IMAGE_TYPE {
    PGM
};

// Delay for a given number of milliseconds
void crossPlatformDelay (int);

```

```
42 // Get the user's desktop path
43 std::string getDesktopPath ();
44
45 // Get information about an equation preset
46 std::string equationPreset (EQ_PRESETS, bool);
47 // Get information about a colour palette preset
48 std::string colourPalettePreset (CP_PRESETS);
49
50 // Get the current working directory from the path argument
51 char* trimExecutableFromPath (char*);
52
53 #endif
```

Testing

In the following test description, some technical elements have been used. Setup statements which begin with ‘./HyperFractal’ signify that the test will consist of running the console version of the application as a command using the arguments specified after the ‘./HyperFractal’.
Expected and Actual Output statements may begin with ‘> …’ which indicates console output occurring. Some tests link to images as evidence, which are shown in the Test Appendices below the testing table.

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
1	1-01	Test available resolution using console interface	./HyperFractal -32 0 0 1 "z^2+c" 4 32	Program exits reporting parameter error.	> Parameter error on argument number 0: > Specified resolution too low.	Pass
2			./HyperFractal 0 0 0 1 "z^2+c" 4 32	Program exits reporting parameter error.	> Parameter error on argument number 0: > Specified resolution too low.	Pass
3			./HyperFractal 32 0 0 1 "z^2+c" 4 32	Program renders the fractal, reports success, and saves a 32x32 image file.	> Rendering with parameters: > Resolution=32 > EvaluationLimit=32 > Threads=4 > Zoom=1.000000e+00 > OffsetX=0.000000000000000000000000000000000000000000000000000000000000000000000000000000 > OffsetY=0.000000000000000000000000000000000000000000000000000000000000000000000000000000 > Rendering done. > Fractal render from 14 Mar 2022 at 10.13.23.507	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
4			./HyperFractal 320 0 0 1 "z^2+c" 4 32	Program renders the fractal, reports success, and saves a 320x320 image file.	> Rendering with parameters: > Resolution=320 > EvaluationLimit=32 > Threads=4 > Zoom=1.000000e+00 > OffsetX=0.000000000000000000000000000000000000000000000000000000000000000000000000000000 > OffsetY=0.000000000000000000000000000000000000000000000000000000000000000000000000000000 > Rendering done. > Fractal render from 14 Mar 2022 at 10.21.43.488	Pass
5			./HyperFractal 3200 0 0 1 "z^2+c" 4 32	Program renders the fractal, reports success, and saves a 3200x3200 image file.	> Rendering with parameters: > Resolution=3200 > EvaluationLimit=32 > Threads=4 > Zoom=1.000000e+00 > OffsetX=0.000000000000000000000000000000000000000000000000000000000000000000000000000000 > OffsetY=0.000000000000000000000000000000000000000000000000000000000000000000000000000000 > Rendering done. > Fractal render from 14 Mar 2022 at 10.22.23.807	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
6			./HyperFractal 32000 0 0 1 "z^2+c" 4 32	Program renders the fractal, reports success, and saves a 32000x32000 image file.	> Rendering with parameters: > Resolution=32000 > EvaluationLimit=32 > Threads=4 > Zoom=1.000000e+00 > OffsetX=0.000000000000000000000000000000000000000000000000000000000000000000000000000000 > OffsetY=0.000000000000000000000000000000000000000000000000000000000000000000000000000000 > Rendering done. > Fractal render from 14 Mar 2022 at 10.23.16.382	Pass
7	1-02	Demonstrate image display in interface	Run in GUI mode and click ‘Render Image’.	Rendered fractal is shown in image panel of interface.	Rendered fractal is shown in image panel of interface.	Pass
8	1-03 2-02	Demonstrate navigating around the interface using buttons	Run in GUI mode and click left navigation button.	Viewport on rendered image shifts to the left.	Viewport on rendered image shifts to the left.	Pass
9			Run in GUI mode and click right navigation button.	Viewport on rendered image shifts to the right.	Viewport on rendered image shifts to the right.	Pass
10			Run in GUI mode and click up navigation button.	Viewport on rendered image shifts up.	Viewport on rendered image shifts up.	Pass
11			Run in GUI mode and click down navigation button.	Viewport on rendered image shifts down.	Viewport on rendered image shifts down.	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
12			Run in GUI mode and click zoom in button.	Viewport on rendered image is scaled up.	Viewport on rendered image is scaled up.	Pass
13			Run in GUI mode and click zoom out button.	Viewport on rendered image is scaled down.	Viewport on rendered image is scaled down.	Pass
14	1-03 2-02	Demonstrate navigating around the interface using keys	Run in GUI mode and press left arrow key.	Viewport on rendered image shifts to the left.	Viewport on rendered image shifts to the left.	Pass
15			Run in GUI mode and press right arrow key.	Viewport on rendered image shifts to the right.	Viewport on rendered image shifts to the right.	Pass
16			Run in GUI mode and press up arrow key.	Viewport on rendered image shifts up.	Viewport on rendered image shifts up.	Pass
17			Run in GUI mode and press down arrow key.	Viewport on rendered image shifts down.	Viewport on rendered image shifts down.	Pass
18			Run in GUI mode and press ‘-’ key.	Viewport on rendered image is scaled up.	Viewport on rendered image is scaled up.	Pass
19			Run in GUI mode and press ‘+’ key.	Viewport on rendered image is scaled down.	Viewport on rendered image is scaled down.	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
20	1-04	Demonstrate equation customisation	Run in GUI mode, click ‘Equation Presets’ and select each one sequentially.	Equation is changed to the selected equation, image is updated accordingly for each one.	Equation is changed to the selected equation, image is updated accordingly for each one.	Pass
21			Run in GUI mode, click on equation input box and type to change equation to ‘1/(zz+ccc)’	Equation input box displays new equation, image is updated to show entered equation.	Equation input box displays new equation, image is updated to show entered equation.	Pass
22	1-04 2-04	Demonstrate equation parsing at runtime	Inputs to HFractalEquationParser::extract_equation(): Valid data: “z^2+c” “(x-y)/z” “((a-x)^2)+((b-y)^2)” “z^c-z” “z^(c-z)” Invalid data: “z^/c” “)z-c)” “iy-c(“ “w” “()+a” “z-.5” “z-3.”	 “z 2 ^ c +” “x y - z /” “a x - 2 ^ b y - 2 ^ +” “z c ^ z -” “z c z - ^” Operation Error Bracket Error Bracket Error Unsupported Character Error Bracket Error Floating Point Error Floating Point Error	 “z 2 ^ c +” “x y - z /” “a x - 2 ^ b y - 2 ^ +” “z c ^ z -” “z c z - ^” Operation Error Bracket Error Bracket Error Unsupported Character Error Bracket Error Floating Point Error Floating Point Error	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
23	1-05	Demonstrate image saving	Run in GUI mode, press ‘Save Image’	Rendered image is saved to a bitmap image on the desktop with a generated name based on the date and time.	Rendered image is saved to a bitmap image on the desktop with a generated name based on the date and time.	Pass
24	1-06 2-06	Demonstrate database saving system	Run in GUI mode, press 'Save Render State’, type a name and press ‘Save’	Console displays ‘Profile saved to database’. Database files created/ have the record appended to them.	Console displays ‘Profile saved to database’. Database has record appended, see Test Appendix 1.	Pass
25		Demonstrate database reading system	Run in GUI mode, press 'Load Render State’, select profile from the displayed list and press ‘Load’. For contents of configuration profile table, see Test Appendix 2.	Console displays ‘Profile loaded from database’. Records stored in the database are converted into rows in the list. When load is pressed, the details of the profile are loaded into the render environment and the image is rerendered.	Console displays ‘Profile loaded from database’. Records are visible in list, see Test Appendix 3. When load is pressed, configuration parameters are copied into rendering environment and the image is rerendered.	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
26	1-07	Timing the rendering process on various different render parameters.	Run in GUI mode, fullscreen application. For each of the following, configure and press ‘Render Image’ and start a timer. Stop the timer when the render reaches 100% completion: Equation and iteration limit: “(z^2)+c”, 200 “(z^2)+c”, 500 “1/((z^2)+c)”, 200 “1/((z^2)+c)”, 500 “z^(c^2)”, 300 “(z^z)+ (c-0.5)”, 300 “z/(zz+ccc)”, 300	For first set of tests, render time should be less than 30 seconds on an average laptop. For the second set, they should be less than 1 minute.	01.35 seconds 02.51 seconds 03.61 seconds 08.00 seconds 19.40 seconds 21.73 seconds 24.48 seconds	Pass
27	1-09 2-08	Demonstrate help and instructions	Run in GUI mode, click ‘Help & Instructions’.	Application displays a help document containing information about how to use the software.	Application displays a help document by opening the project’s README in the native system web browser. See Appendix 1 for this help document.	Pass
28	1-10	Demonstrate changing render parameters using interface buttons	Run in GUI mode, click ‘>’ button to increase iteration limit.	Iteration limit is increased by one, iteration limit display box is updated.	Iteration limit is increased by one, iteration limit display box is updated.	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
29			Run in GUI mode, click ‘<’ button to decrease iteration limit.	Iteration limit is decreased by one, iteration limit display box is updated.	Iteration limit is decreased by one, iteration limit display box is updated.	Pass
30			Run in GUI mode, click ‘Colour Palettes’. Select each one sequentially.	Colour palette of displayed image changes when a colour palette is selected, for each one.	Colour palette of displayed image changes to the selected palette when a colour palette is selected, for each one.	Pass
31		Demonstrate changing render parameters using keyboard keys	Run in GUI mode, press ‘]’ key to. Repeat, holding shift.	Iteration limit is increased by one, iteration limit display box is updated. Iteration limit is increased by ten, iteration limit display box is updated.	Iteration limit is increased by one, iteration limit display box is updated. Iteration limit is increased by ten, iteration limit display box is updated.	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
32			Run in GUI mode, press '[' key to. Repeat, holding shift.	Iteration limit is decreased by one, iteration limit display box is updated. Iteration limit is decreased by ten, iteration limit display box is updated.	Iteration limit is decreased by one, iteration limit display box is updated. Iteration limit is decreased by ten, iteration limit display box is updated.	Pass
33	1-11 2-03	Demonstrate the application running on both macOS and Windows.	On macOS: Double-click the packaged application. Open a terminal, navigate to the executable, run "./ Hyperfractal" Do the same, running "./ Hyperfractal 256 0.0 0.0 1.0 "z^2+c" 4 200"	The application will open in GUI mode. The application will open in GUI mode. Program runs in terminal, performs the render, saves the image to the desktop.	Application opens in GUI mode. Application opens in GUI mode. Program runs in terminal, performs the render, saves the image to the desktop.	Pass
34			On Windows: Double-click the packaged application.	The application will open in GUI mode.	Application opens in GUI mode.	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
35	2-01	Demonstrate preview render	Run in GUI mode. Press ‘Render Image’. Perform any navigation event. Press ‘Render Image’.	Preview render shown. Full render occurs and is shown Preview render switched to. Full render occurs and is shown.	Preview render shown. Full render occurs and is shown Preview render switched to. Full render occurs and is shown. See Test Appendix 4 for a screenshot after step 3, and Test Appendix 5 for a screenshot after step 4.	Pass
36	2-05	Demonstrate detection of available threads.	Run in CLI mode, specifying the number of threads to be used (here 4): ./HyperFractal 1024 0.0 0.0 0.75 "(z^2)+c" 4 150	Program displays details of render environment before rendering. Activity monitor shows process using 5 threads (4 workers + 1 main).	> Rendering with parameters: > Resolution=1024 > EvaluationLimit=150 > Threads=4 > Zoom=7.500000e-01 > OffsetX=0.0000000000000000000000000000000000000000000000000000000000000000000000000000 > OffsetY=0.0000000000000000000000000000000000000000000000000000000000000000000000000000 > Rendering done. Activity monitor shows executable process is using 5 threads.	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
37		Demonstrate performance boost using multiple threads.	Run in CLI mode with only 1 worker: ./HyperFractal 2048 0.0 0.0 0.75 "(z^2)+c" 1 500 ./HyperFractal 2048 0.0 0.0 0.75 “(z^2)+c" 2 500 ./HyperFractal 2048 0.0 0.0 0.75 “(z^2)+c" 4 500 ./HyperFractal 2048 0.0 0.0 0.75 “(z^2)+c" 8 500	First run will be the slowest, the third the fastest, the second and fourth somewhere between. Test host machine has 4 cores, hence the prediction.	First run: 10.86 seconds Second run: 7.08 seconds Third run: 5.56 seconds Fourth run: 5.79 seconds As expected, the optimal number of threads for best performance is the number of CPU cores in the host machine.	Pass
38	2-07	Demonstrate resizing ability	Run in GUI mode, drag corner to resize window. Press ‘Render Image’ and resize the window while the render occurs.	Render image frame resizes with window. Window snaps back to original size without image resizing.	Render image frame resizes with window. Window snaps back to original size without image resizing.	Pass
39	2-09	Demonstrate memory requirement	Run in GUI mode, enter fullscreen. Run 20 renders. Leave application open overnight.	Memory requirement is constant over time, and remains less than 500MiB.	Initial requirement: ~30MiB After 20 renders: ~30MiB After being left: ~30MiB Memory requirement is minimal and consistent.	Pass

Test	Req.	Test Description	Test Data/ Setup	Expected Output/ Result	Actual Output/Result	Test Result
40	2-10	Demonstrate render progress	Run in GUI mode, press ‘Render Image’.	Console text changes to “Rendering…”. Progress bar shows render progress. Upon completion, console text changes to “Rendering done.”.	Console text changes to “Rendering…”. Progress bar shows render progress. Upon completion, console text changes to “Rendering done.”.	Pass
41	2-11	Demonstrate parameter locking during render	Run in GUI mode and press ‘Render Image’. While render is happening, click zoom in, zoom out, increase/ decrease iteration limit, and navigate up/down/left/ right. Press all navigation keys on the keyboard. Click inside the render area.	GUI should not respond to any inputs (navigation should not occur, iteration limit and zoom should be kept the same).	GUI does not respond to any inputs, all parameters are kept the same.	Pass
42	Misc	Test custom HSV to RGB conversion function	Feed in sets of three inputs to the function. Test: H = from 0 to 1 in steps of 1/6 S = from 0 to 1 in steps of 0.1 V = from 0 to 1 in steps of 0.1 In every combination.	Function should output a 32bit integer representing the RGBA colour generated.	Function outputs a fully correct test set (compared with a commercially available HSV to RGB converter). See Test Appendix 5 for results.	Pass

Summary

The tests above have targeted all of the requirements for this project, except those which require user feedback (which are discussed below). Every test has successfully passed, proving that all of my high-level and low-level objectives (except for 1-08) have been met by the technical solution I developed.

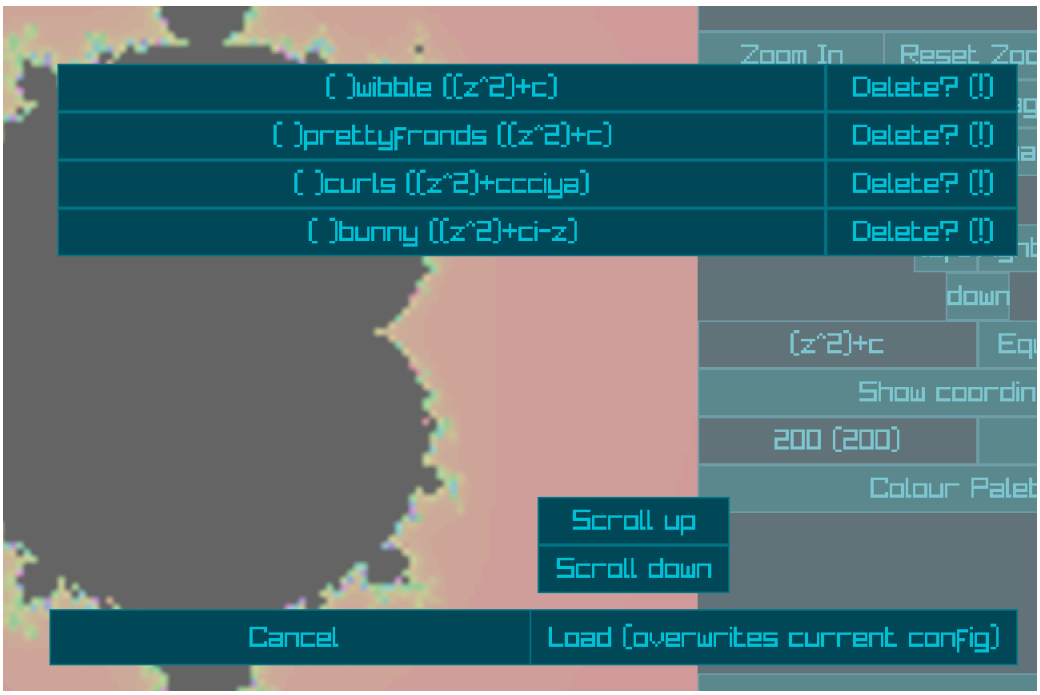
Test Appendices

4	0.000000	0.000000	1.000000	200	(z^2)+c	deletethis	2	0
0	-0.105455	0.909091	86.497559	200	(z^2)+c	wibble	1	0
2	0.439259	0.333015	3325.256730	200	(z^2)+c	prettyfronds	1	0
1	0.708664	-1.054763	5.062500	100	(z^2)+ccciya	curls	1	0
3	0.140720	-0.308651	1.000000	160	(z^2)+ci-z	bunny	2	0

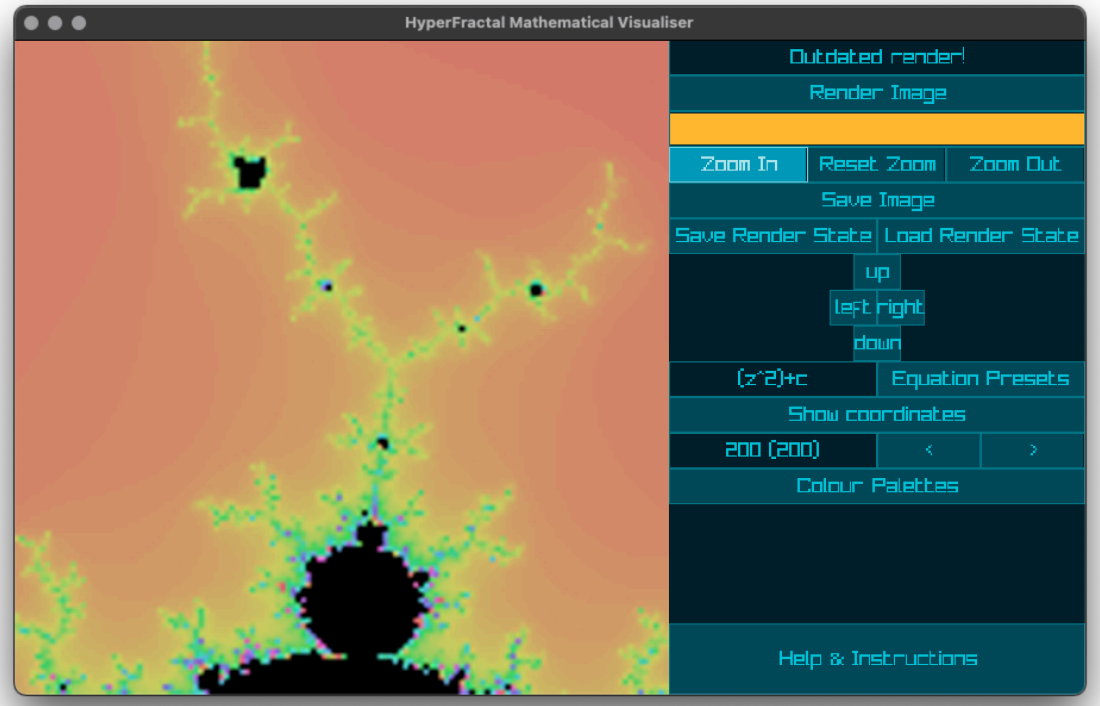
Test Appendix 1

0	-0.105455	0.909091	86.497559	200	(z^2)+c	wibble	1	0
2	0.439259	0.333015	3325.256730	200	(z^2)+c	prettyfronds	1	0
1	0.708664	-1.054763	5.062500	100	(z^2)+ccciya	curls	1	0
3	0.140720	-0.308651	1.000000	160	(z^2)+ci-z	bunny	2	0

Test Appendix 2



Test Appendix 3



Test Appendix 4

Test Appendix 5

The code used for this test is below, while the results are on the following pages.

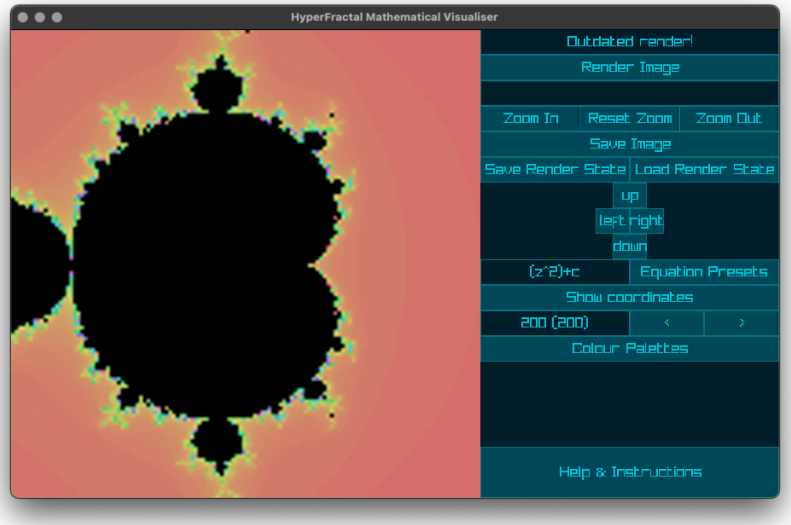
```
bool success = true;
cout << hex;
for (float f = 0; f < 1; f+=(1.0/6.0)) {
    for (float s = 0; s < 1; s+=0.1) {
        for (float v = 0; v < 1; v+=0.1) {
            int my_colour = HFractalImage::HSVToRGB (f, s, 1);
            int raylib_colour = ColorToInt(ColorFromHSV(f*360, s, 1));
            if (my_colour != raylib_colour) success = false;
            cout << my_colour << " " << raylib_colour << ((my_colour !=
raylib_colour) ? " FAIL" : " PASS") << endl;
        }
    }
}

if (success) {
    cout << "Test succeeded :D" << endl;
} else {
    cout << "Test failed :(" << endl;
}
```

```
ff ff PASS
191919ff 191919ff PASS
333333ff 333333ff PASS
4c4c4cff 4c4c4cff PASS
666666ff 666666ff PASS
7f7f7fff 7f7f7fff PASS
999999ff 999999ff PASS
b2b2b2ff b2b2b2ff PASS
ccccccff ccccccff PASS
e5e5e5ff e5e5e5ff PASS
ff ff PASS
191616ff 191616ff PASS
332d2dff 332d2dff PASS
4c4444ff 4c4444ff PASS
665b5bff 665b5bff PASS
7f7272ff 7f7272ff PASS
998989ff 998989ff PASS
b2a0a0ff b2a0a0ff PASS
ccb7b7ff ccb7b7ff PASS
e5ceceff e5ceceff PASS
ff ff PASS
191414ff 191414ff PASS
332828ff 332828ff PASS
4c3d3dff 4c3d3dff PASS
665151ff 665151ff PASS
7f6666ff 7f6666ff PASS
997a7aff 997a7aff PASS
b28e8eff b28e8eff PASS
cca3a3ff cca3a3ff PASS
e5b7b7ff e5b7b7ff PASS
ff ff PASS
191111ff 191111ff PASS
332323ff 332323ff PASS
4c3535ff 4c3535ff PASS
664747ff 664747ff PASS
7f5959ff 7f5959ff PASS
996b6bff 996b6bff PASS
b27c7cff b27c7cff PASS
cc8e8eff cc8e8eff PASS
e5a0a0ff e5a0a0ff PASS
ff ff PASS
190f0fff 190f0fff PASS
331e1eff 331e1eff PASS
4c2d2dff 4c2d2dff PASS
663d3dff 663d3dff PASS
7f4c4cff 7f4c4cff PASS
995b5bff 995b5bff PASS
b26b6bff b26b6bff PASS
cc7a7aff cc7a7aff PASS
e58989ff e58989ff PASS
ff ff PASS
190c0cff 190c0cff PASS
331919ff 331919ff PASS
4c2626ff 4c2626ff PASS
663333ff 663333ff PASS
7f3f3fff 7f3f3fff PASS
994c4cff 994c4cff PASS
b25959ff b25959ff PASS
cc6666ff cc6666ff PASS
e57272ff e57272ff PASS
ff ff PASS
190a0aff 190a0aff PASS
331414ff 331414ff PASS
4c1e1eff 4c1e1eff PASS
662828ff 662828ff PASS
7f3232ff 7f3232ff PASS
993d3dff 993d3dff PASS
b24747ff b24747ff PASS
cc5151ff cc5151ff PASS
e55b5bff e55b5bff PASS
ff ff PASS
190707ff 190707ff PASS
330f0fff 330f0fff PASS
4c1616ff 4c1616ff PASS
661e1eff 661e1eff PASS
7f2626ff 7f2626ff PASS
992d2dff 992d2dff PASS
b23535ff b23535ff PASS
cc3d3dff cc3d3dff PASS
e54444ff e54444ff PASS
ff ff PASS
190505ff 190505ff PASS
330a0aff 330a0aff PASS
4c0f0fff 4c0f0fff PASS
661414ff 661414ff PASS
7f1919ff 7f1919ff PASS
991e1eff 991e1eff PASS
b22323ff b22323ff PASS
cc2828ff cc2828ff PASS
e52d2dff e52d2dff PASS
ff ff PASS
190202ff 190202ff PASS
330505ff 330505ff PASS
4c0707ff 4c0707ff PASS
660a0aff 660a0aff PASS
7f0c0cff 7f0c0cff PASS
990f0fff 990f0fff PASS
b21111ff b21111ff PASS
cc1414ff cc1414ff PASS
e51616ff e51616ff PASS
ff ff PASS
191919ff 191919ff PASS
333333ff 333333ff PASS
4c4c4cff 4c4c4cff PASS
666666ff 666666ff PASS
7f7f7fff 7f7f7fff PASS
999999ff 999999ff PASS
b2b2b2ff b2b2b2ff PASS
ccccccff ccccccff PASS
e5e5e5ff e5e5e5ff PASS
ff ff PASS
191916ff 191916ff PASS
33332dff 33332dff PASS
4c4c44ff 4c4c44ff PASS
66665bff 66665bff PASS
7f7f72ff 7f7f72ff PASS
999989ff 999989ff PASS
b2b2a0ff b2b2a0ff PASS
ccccb7ff ccccb7ff PASS
e5e5ceff e5e5ceff PASS
ff ff PASS
191914ff 191914ff PASS
333328ff 333328ff PASS
4c4c3dff 4c4c3dff PASS
666651ff 666651ff PASS
7f7f66ff 7f7f66ff PASS
99997aff 99997aff PASS
b2b28eff b2b28eff PASS
cccca3ff cccca3ff PASS
e5e5b7ff e5e5b7ff PASS
ff ff PASS
191911ff 191911ff PASS
333323ff 333323ff PASS
4c4c35ff 4c4c35ff PASS
666647ff 666647ff PASS
7f7f59ff 7f7f59ff PASS
99996bff 99996bff PASS
b2b27cff b2b27cff PASS
cccc8eff cccc8eff PASS
e5e5a0ff e5e5a0ff PASS
ff ff PASS
19190fff 19190fff PASS
33331eff 33331eff PASS
4c4c2dff 4c4c2dff PASS
66663dff 66663dff PASS
7f7f4cff 7f7f4cff PASS
99995bff 99995bff PASS
b2b26bff b2b26bff PASS
cccc7aff cccc7aff PASS
e5e589ff e5e589ff PASS
ff ff PASS
19190cff 19190cff PASS
333319ff 333319ff PASS
4c4c26ff 4c4c26ff PASS
666633ff 666633ff PASS
7f7f3fff 7f7f3fff PASS
99994cff 99994cff PASS
b2b259ff b2b259ff PASS
cccc66ff cccc66ff PASS
e5e572ff e5e572ff PASS
ff ff PASS
19190aff 19190aff PASS
333314ff 333314ff PASS
4c4c1eff 4c4c1eff PASS
666628ff 666628ff PASS
7f7f32ff 7f7f32ff PASS
99993dff 99993dff PASS
b2b247ff b2b247ff PASS
cccc51ff cccc51ff PASS
e5e55bff e5e55bff PASS
ff ff PASS
191907ff 191907ff PASS
33330fff 33330fff PASS
4c4c16ff 4c4c16ff PASS
66661eff 66661eff PASS
7f7f26ff 7f7f26ff PASS
99992dff 99992dff PASS
b2b235ff b2b235ff PASS
cccc3dff cccc3dff PASS
e5e544ff e5e544ff PASS
ff ff PASS
191905ff 191905ff PASS
33330aff 33330aff PASS
4c4c0fff 4c4c0fff PASS
666614ff 666614ff PASS
7f7f19ff 7f7f19ff PASS
99991eff 99991eff PASS
b2b223ff b2b223ff PASS
cccc28ff cccc28ff PASS
e5e52dff e5e52dff PASS
ff ff PASS
191902ff 191902ff PASS
333305ff 333305ff PASS
4c4c07ff 4c4c07ff PASS
66660aff 66660aff PASS
7f7f0cff 7f7f0cff PASS
99990fff 99990fff PASS
b2b211ff b2b211ff PASS
cccc14ff cccc14ff PASS
e5e516ff e5e516ff PASS
ff ff PASS
191919ff 191919ff PASS
333333ff 333333ff PASS
4c4c4cff 4c4c4cff PASS
666666ff 666666ff PASS
7f7f7fff 7f7f7fff PASS
999999ff 999999ff PASS
b2b2b2ff b2b2b2ff PASS
cccccccff cccccccff PASS
e5e5e5ff e5e5e5ff PASS
ff ff PASS
161916ff 161916ff PASS
2d332dff 2d332dff PASS
444c44ff 444c44ff PASS
5b665bff 5b665bff PASS
727f72ff 727f72ff PASS
899989ff 899989ff PASS
a0b2a0ff a0b2a0ff PASS
b7ccb7ff b7ccb7ff PASS
cee5ceff cee5ceff PASS
ff ff PASS
141914ff 141914ff PASS
283328ff 283328ff PASS
3d4c3dff 3d4c3dff PASS
516651ff 516651ff PASS
667f66ff 667f66ff PASS
7a997aff 7a997aff PASS
8eb28eff 8eb28eff PASS
a3cca3ff a3cca3ff PASS
b7e5b7ff b7e5b7ff PASS
ff ff PASS
111911ff 111911ff PASS
233323ff 233323ff PASS
354c35ff 354c35ff PASS
476647ff 476647ff PASS
597f59ff 597f59ff PASS
6b996bff 6b996bff PASS
7cb27cff 7cb27cff PASS
8ecc8eff 8ecc8eff PASS
a0e5a0ff a0e5a0ff PASS
ff ff PASS
f190fff f190fff PASS
1e331eff 1e331eff PASS
2d4c2dff 2d4c2dff PASS
3d663dff 3d663dff PASS
4c7f4cff 4c7f4cff PASS
5b995bff 5b995bff PASS
6bb26bff 6bb26bff PASS
7acc7aff 7acc7aff PASS
89e589ff 89e589ff PASS
ff ff PASS
c190cff c190cff PASS
193319ff 193319ff PASS
264c26ff 264c26ff PASS
336633ff 336633ff PASS
3f7f3fff 3f7f3fff PASS
4c994cff 4c994cff PASS
59b259ff 59b259ff PASS
66cc66ff 66cc66ff PASS
72e572ff 72e572ff PASS
ff ff PASS
a190aff a190aff PASS
143314ff 143314ff PASS
1e4c1eff 1e4c1eff PASS
286628ff 286628ff PASS
327f32ff 327f32ff PASS
3d993dff 3d993dff PASS
47b247ff 47b247ff PASS
51cc51ff 51cc51ff PASS
5be55bff 5be55bff PASS
ff ff PASS
71907ff 71907ff PASS
f330fff f330fff PASS
164c16ff 164c16ff PASS
1e661eff 1e661eff PASS
267f26ff 267f26ff PASS
2d992dff 2d992dff PASS
35b235ff 35b235ff PASS
3dcc3dff 3dcc3dff PASS
44e544ff 44e544ff PASS
ff ff PASS
51905ff 51905ff PASS
a330aff a330aff PASS
f4c0fff f4c0fff PASS
146614ff 146614ff PASS
197f19ff 197f19ff PASS
1e991eff 1e991eff PASS
23b223ff 23b223ff PASS
28cc28ff 28cc28ff PASS
2de52dff 2de52dff PASS
ff ff PASS
21902ff 21902ff PASS
53305ff 53305ff PASS
74c07ff 74c07ff PASS
a660aff a660aff PASS
c7f0cff c7f0cff PASS
f990fff f990fff PASS
11b211ff 11b211ff PASS
14cc14ff 14cc14ff PASS
16e516ff 16e516ff PASS
ff ff PASS
191919ff 191919ff PASS
333333ff 333333ff PASS
4c4c4cff 4c4c4cff PASS
666666ff 666666ff PASS
7f7f7fff 7f7f7fff PASS
999999ff 999999ff PASS
b2b2b2ff b2b2b2ff PASS
ccccccff ccccccff PASS
e5e5e5ff e5e5e5ff PASS
ff ff PASS
161919ff 161919ff PASS
2d3333ff 2d3333ff PASS
444c4cff 444c4cff PASS
5b6666ff 5b6666ff PASS
727f7fff 727f7fff PASS
899999ff 899999ff PASS
a0b2b2ff a0b2b2ff PASS
b7ccccff b7ccccff PASS
cee5e5ff cee5e5ff PASS
ff ff PASS
141919ff 141919ff PASS
283333ff 283333ff PASS
3d4c4cff 3d4c4cff PASS
516666ff 516666ff PASS
667f7fff 667f7fff PASS
7a9999ff 7a9999ff PASS
8eb2b2ff 8eb2b2ff PASS
a3ccccff a3ccccff PASS
b7e5e5ff b7e5e5ff PASS
ff ff PASS
111919ff 111919ff PASS
233333ff 233333ff PASS
354c4cff 354c4cff PASS
476666ff 476666ff PASS
597f7fff 597f7fff PASS
6b9999ff 6b9999ff PASS
7cb2b2ff 7cb2b2ff PASS
8eccccff 8eccccff PASS
a0e5e5ff a0e5e5ff PASS
```

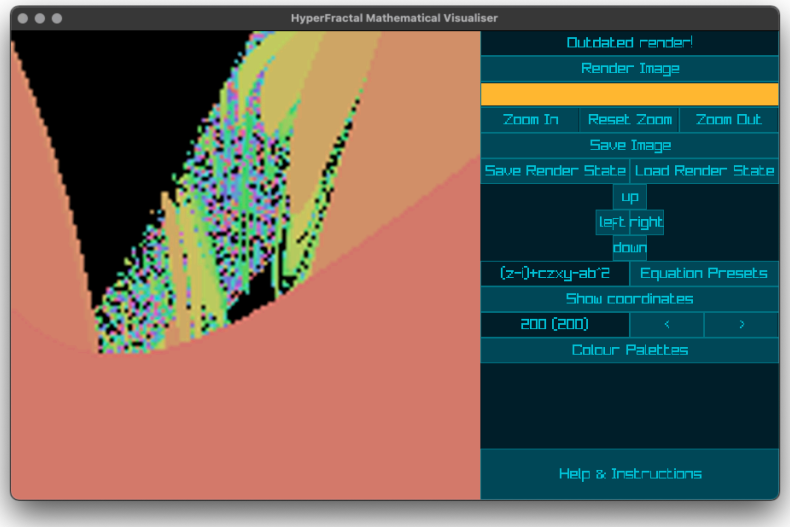
```
ff ff PASS
a1919ff a1919ff PASS
143333ff 143333ff PASS
1e4c4cff 1e4c4cff PASS
286666ff 286666ff PASS
327f7fff 327f7fff PASS
3d9999ff 3d9999ff PASS
47b2b2ff 47b2b2ff PASS
51ccccff 51ccccff PASS
5be5e5ff 5be5e5ff PASS
ff ff PASS
71919ff 71919ff PASS
f3333ff f3333ff PASS
164c4cff 164c4cff PASS
1e6666ff 1e6666ff PASS
267f7fff 267f7fff PASS
2d9999ff 2d9999ff PASS
35b2b2ff 35b2b2ff PASS
3dccccff 3dccccff PASS
44e5e5ff 44e5e5ff PASS
ff ff PASS
51919ff 51919ff PASS
a3333ff a3333ff PASS
f4c4cff f4c4cff PASS
146666ff 146666ff PASS
197f7fff 197f7fff PASS
1e9999ff 1e9999ff PASS
23b2b2ff 23b2b2ff PASS
28ccccff 28ccccff PASS
2de5e5ff 2de5e5ff PASS
ff ff PASS
21919ff 21919ff PASS
53333ff 53333ff PASS
74c4cff 74c4cff PASS
a6666ff a6666ff PASS
c7f7fff c7f7fff PASS
f9999ff f9999ff PASS
11b2b2ff 11b2b2ff PASS
14ccccff 14ccccff PASS
16e5e5ff 16e5e5ff PASS
ff ff PASS
191919ff 191919ff PASS
333333ff 333333ff PASS
4c4c4cff 4c4c4cff PASS
666666ff 666666ff PASS
7f7f7fff 7f7f7fff PASS
999999ff 999999ff PASS
b2b2b2ff b2b2b2ff PASS
cccccff cccccff PASS
e5e5e5ff e5e5e5ff PASS
ff ff PASS
161619ff 161619ff PASS
2d2d33ff 2d2d33ff PASS
44444cff 44444cff PASS
5b5b66ff 5b5b66ff PASS
72727fff 72727fff PASS
898999ff 898999ff PASS
a0a0b2ff a0a0b2ff PASS
b7b7ccff b7b7ccff PASS
cecee5ff cecee5ff PASS
ff ff PASS
141419ff 141419ff PASS
282833ff 282833ff PASS
3d3d4cff 3d3d4cff PASS
515166ff 515166ff PASS
66667fff 66667fff PASS
7a7a99ff 7a7a99ff PASS
8e8eb2ff 8e8eb2ff PASS
a3a3ccff a3a3ccff PASS
b7b7e5ff b7b7e5ff PASS
ff ff PASS
111119ff 111119ff PASS
232333ff 232333ff PASS
35354cff 35354cff PASS
474766ff 474766ff PASS
59597fff 59597fff PASS
6b6b99ff 6b6b99ff PASS
7c7cb2ff 7c7cb2ff PASS
8e8eccff 8e8eccff PASS
a0a0e5ff a0a0e5ff PASS
ff ff PASS
f0f19ff f0f19ff PASS
1e1e33ff 1e1e33ff PASS
2d2d4cff 2d2d4cff PASS
3d3d66ff 3d3d66ff PASS
4c4c7fff 4c4c7fff PASS
5b5b99ff 5b5b99ff PASS
6b6bb2ff 6b6bb2ff PASS
7a7accff 7a7accff PASS
8989e5ff 8989e5ff PASS
ff ff PASS
c0c19ff c0c19ff PASS
191933ff 191933ff PASS
26264cff 26264cff PASS
333366ff 333366ff PASS
3f3f7fff 3f3f7fff PASS
4c4c99ff 4c4c99ff PASS
5959b2ff 5959b2ff PASS
6666ccff 6666ccff PASS
7272e5ff 7272e5ff PASS
ff ff PASS
a0a19ff a0a19ff PASS
141433ff 141433ff PASS
1e1e4cff 1e1e4cff PASS
282866ff 282866ff PASS
32327fff 32327fff PASS
3d3d99ff 3d3d99ff PASS
4747b2ff 4747b2ff PASS
5151ccff 5151ccff PASS
5b5be5ff 5b5be5ff PASS
ff ff PASS
70719ff 70719ff PASS
f0f33ff f0f33ff PASS
16164cff 16164cff PASS
1e1e66ff 1e1e66ff PASS
26267fff 26267fff PASS
2d2d99ff 2d2d99ff PASS
3535b2ff 3535b2ff PASS
3d3dccff 3d3dccff PASS
4444e5ff 4444e5ff PASS
ff ff PASS
50519ff 50519ff PASS
a0a33ff a0a33ff PASS
f0f4cff f0f4cff PASS
141466ff 141466ff PASS
19197fff 19197fff PASS
1e1e99ff 1e1e99ff PASS
2323b2ff 2323b2ff PASS
2828ccff 2828ccff PASS
2d2de5ff 2d2de5ff PASS
ff ff PASS
20219ff 20219ff PASS
50533ff 50533ff PASS
7074cff 7074cff PASS
a0a66ff a0a66ff PASS
c0c7fff c0c7fff PASS
f0f99ff f0f99ff PASS
1111b2ff 1111b2ff PASS
1414ccff 1414ccff PASS
1616e5ff 1616e5ff PASS
ff ff PASS
191919ff 191919ff PASS
333333ff 333333ff PASS
4c4c4cff 4c4c4cff PASS
666666ff 666666ff PASS
7f7f7fff 7f7f7fff PASS
999999ff 999999ff PASS
b2b2b2ff b2b2b2ff PASS
cccccff cccccff PASS
e5e5e5ff e5e5e5ff PASS
ff ff PASS
191619ff 191619ff PASS
332d33ff 332d33ff PASS
4c444cff 4c444cff PASS
665b66ff 665b66ff PASS
7f727fff 7f727fff PASS
998999ff 998999ff PASS
b2a0b2ff b2a0b2ff PASS
ccb7ccff ccb7ccff PASS
e5cee5ff e5cee5ff PASS
ff ff PASS
191419ff 191419ff PASS
332833ff 332833ff PASS
4c3d4cff 4c3d4cff PASS
665166ff 665166ff PASS
7f667fff 7f667fff PASS
997a99ff 997a99ff PASS
b28eb2ff b28eb2ff PASS
cca3ccff cca3ccff PASS
e5b7e5ff e5b7e5ff PASS
ff ff PASS
191119ff 191119ff PASS
332333ff 332333ff PASS
4c354cff 4c354cff PASS
664766ff 664766ff PASS
7f597fff 7f597fff PASS
996b99ff 996b99ff PASS
b27cb2ff b27cb2ff PASS
cc8eccff cc8eccff PASS
e5a0e5ff e5a0e5ff PASS
ff ff PASS
190f19ff 190f19ff PASS
331e33ff 331e33ff PASS
4c2d4cff 4c2d4cff PASS
663d66ff 663d66ff PASS
7f4c7fff 7f4c7fff PASS
995b99ff 995b99ff PASS
b26bb2ff b26bb2ff PASS
cc7accff cc7accff PASS
e589e5ff e589e5ff PASS
ff ff PASS
190c19ff 190c19ff PASS
331933ff 331933ff PASS
4c264cff 4c264cff PASS
663366ff 663366ff PASS
7f3f7fff 7f3f7fff PASS
994c99ff 994c99ff PASS
b259b2ff b259b2ff PASS
cc66ccff cc66ccff PASS
e572e5ff e572e5ff PASS
ff ff PASS
190a19ff 190a19ff PASS
331433ff 331433ff PASS
4c1e4cff 4c1e4cff PASS
662866ff 662866ff PASS
7f327fff 7f327fff PASS
993d99ff 993d99ff PASS
b247b2ff b247b2ff PASS
cc51ccff cc51ccff PASS
e55be5ff e55be5ff PASS
ff ff PASS
190719ff 190719ff PASS
330f33ff 330f33ff PASS
4c164cff 4c164cff PASS
661e66ff 661e66ff PASS
7f267fff 7f267fff PASS
992d99ff 992d99ff PASS
b235b2ff b235b2ff PASS
cc3dccff cc3dccff PASS
e544e5ff e544e5ff PASS
ff ff PASS
190519ff 190519ff PASS
330a33ff 330a33ff PASS
4c0f4cff 4c0f4cff PASS
661466ff 661466ff PASS
7f197fff 7f197fff PASS
991e99ff 991e99ff PASS
b223b2ff b223b2ff PASS
cc28ccff cc28ccff PASS
e52de5ff e52de5ff PASS
ff ff PASS
190219ff 190219ff PASS
330533ff 330533ff PASS
4c074cff 4c074cff PASS
660a66ff 660a66ff PASS
7f0c7fff 7f0c7fff PASS
990f99ff 990f99ff PASS
b211b2ff b211b2ff PASS
cc14ccff cc14ccff PASS
e516e5ff e516e5ff PASS
ff ff PASS
f1919ff f1919ff PASS
1e3333ff 1e3333ff PASS
2d4c4cff 2d4c4cff PASS
3d6666ff 3d6666ff PASS
4c7f7fff 4c7f7fff PASS
5b9999ff 5b9999ff PASS
6bb2b2ff 6bb2b2ff PASS
7accccff 7accccff PASS
89e5e5ff 89e5e5ff PASS
ff ff PASS
c1919ff c1919ff PASS
193333ff 193333ff PASS
264c4cff 264c4cff PASS
336666ff 336666ff PASS
3f7f7fff 3f7f7fff PASS
4c9999ff 4c9999ff PASS
59b2b2ff 59b2b2ff PASS
66ccccff 66ccccff PASS
72e5e5ff 72e5e5ff PASS
```

Evaluation

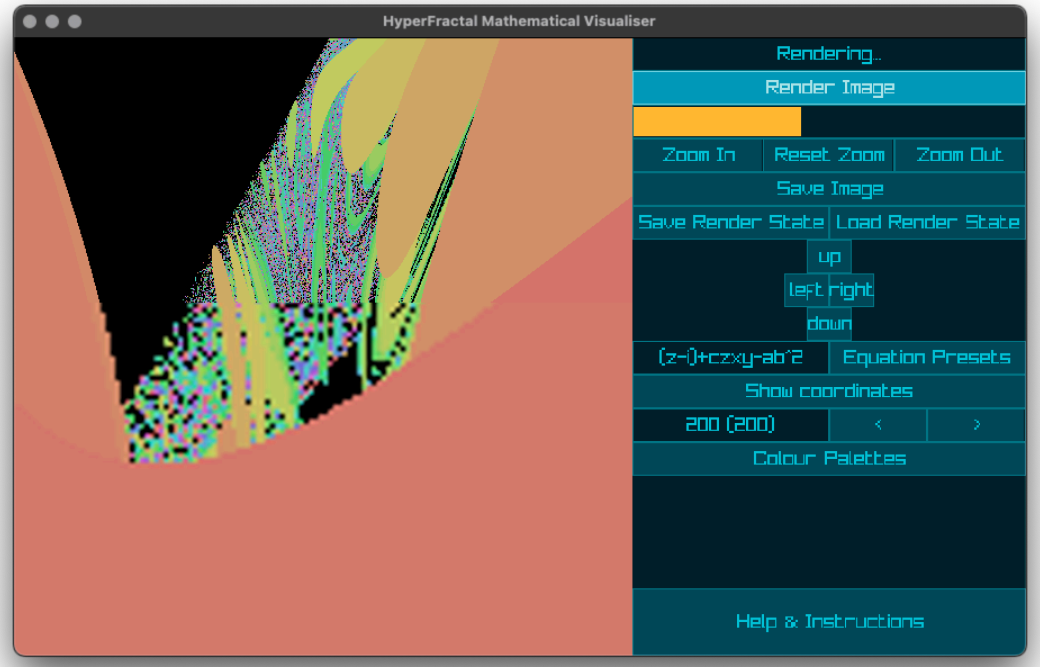
Having targeted each of my high-level requirements already during the testing process, the primary focus of my evaluation of this project will be demonstrating and reviewing the usability of it by interviewing my user and demonstrating the final application working. Below are screenshots showing the application working, using most of its features including saving and loading render states, rendering images, zooming in and out, navigating, and changing the equation.



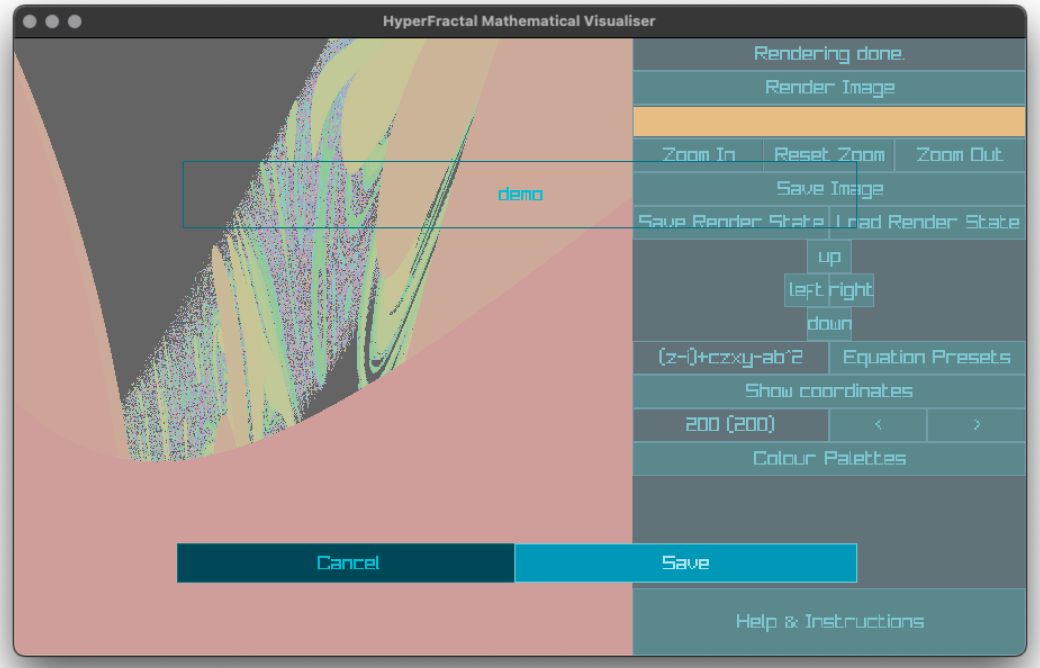
1. Immediately after starting the application, the preview render is shown using default parameters (the Mandelbrot Set, 200 iterations, zoom=0.75, x offset=0, y offset=0).



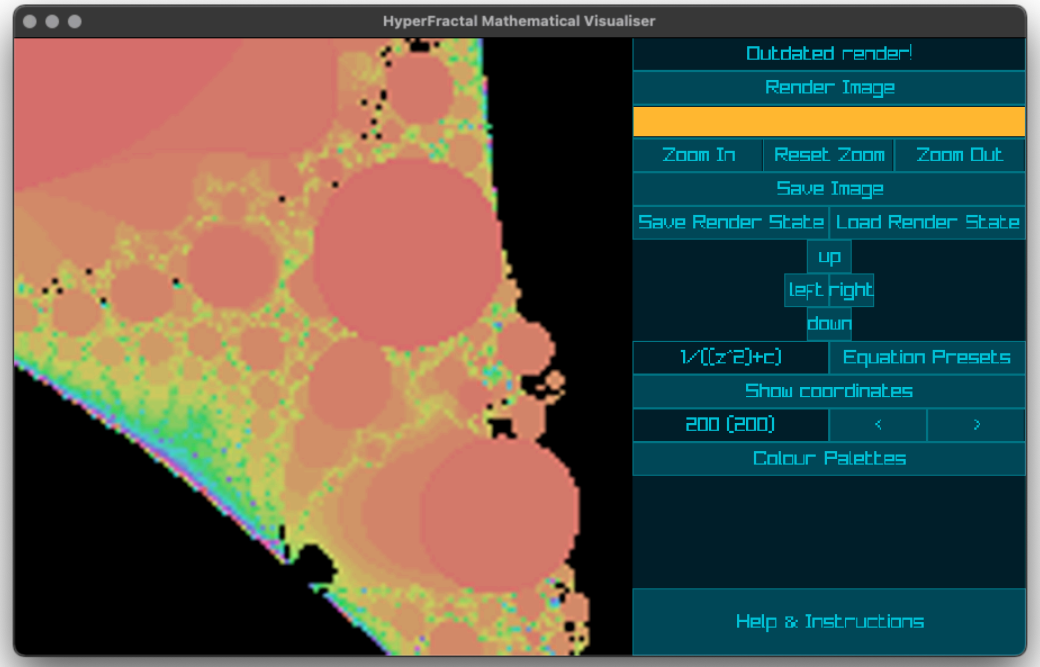
2. Typing in a custom equation to the application, the preview render is shown until the ‘Render Image’ button is pressed



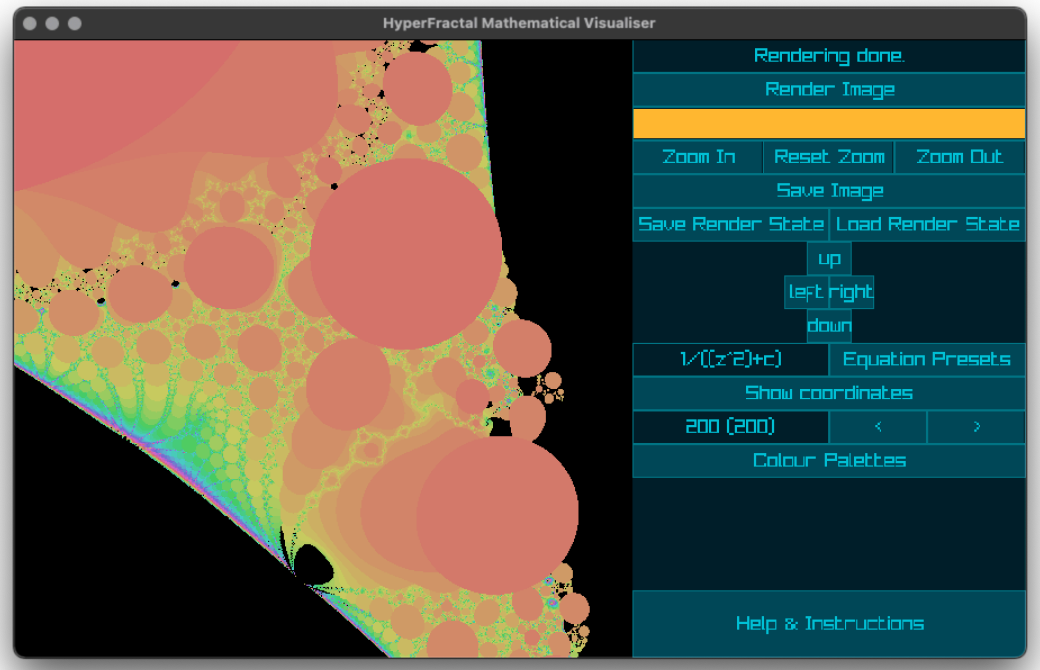
3. The application mid-render, after pressing the ‘Render Image’ button



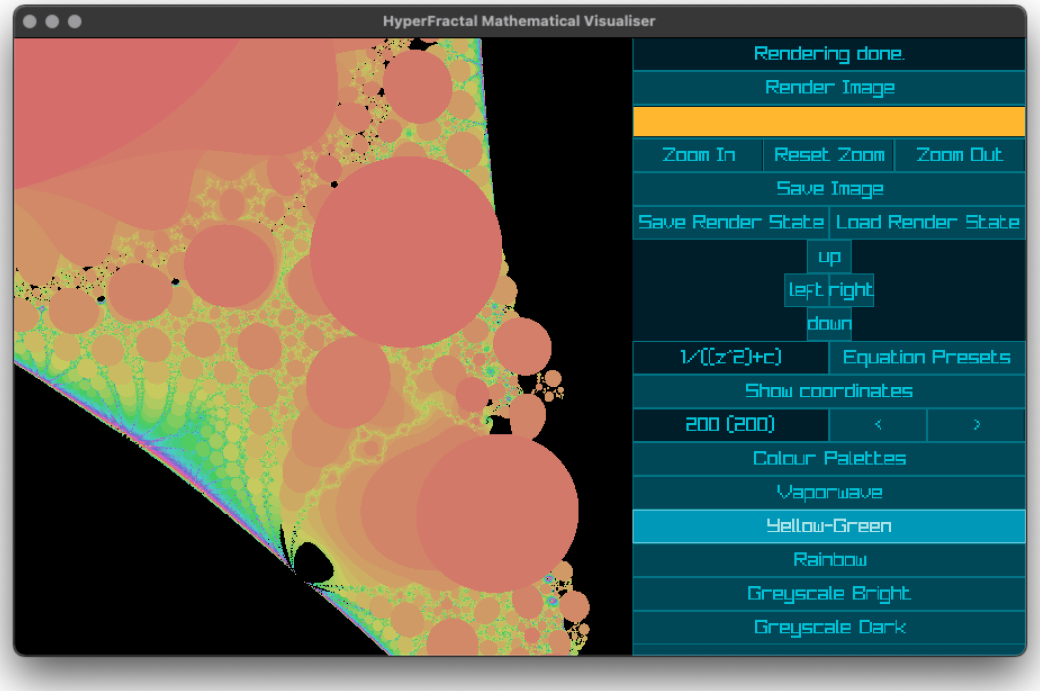
4. After clicking the ‘Save Render State’ button, we save the current render parameters, including the custom equation, under the name ‘demo’



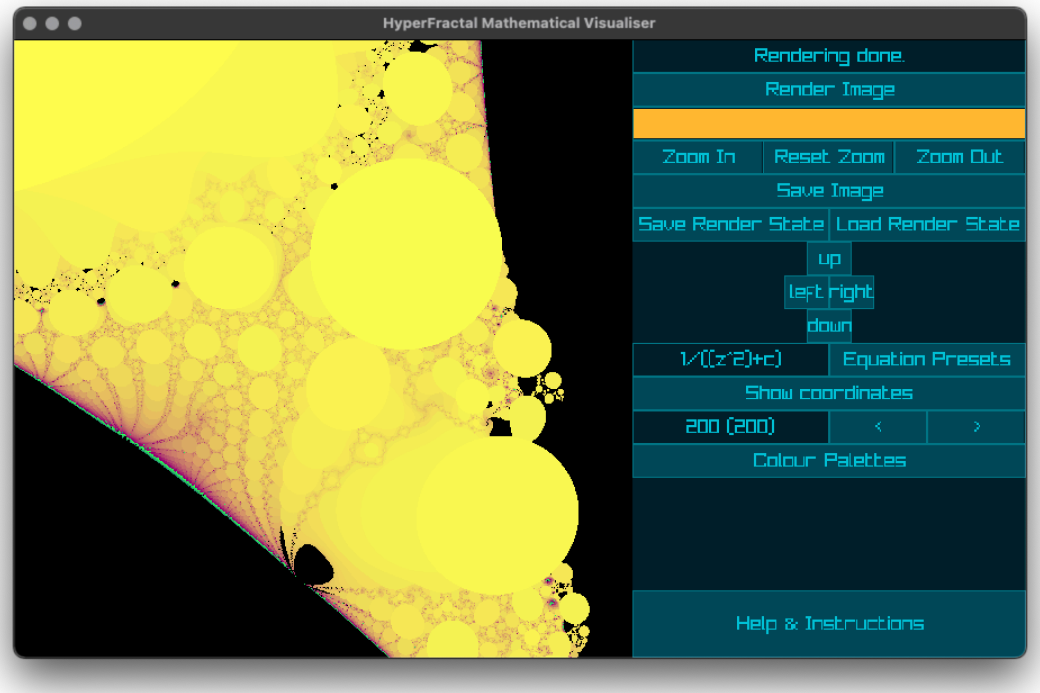
5. The equation changes after selecting the ‘Reciprocal’ equation from the ‘Equation Presets’ dropdown



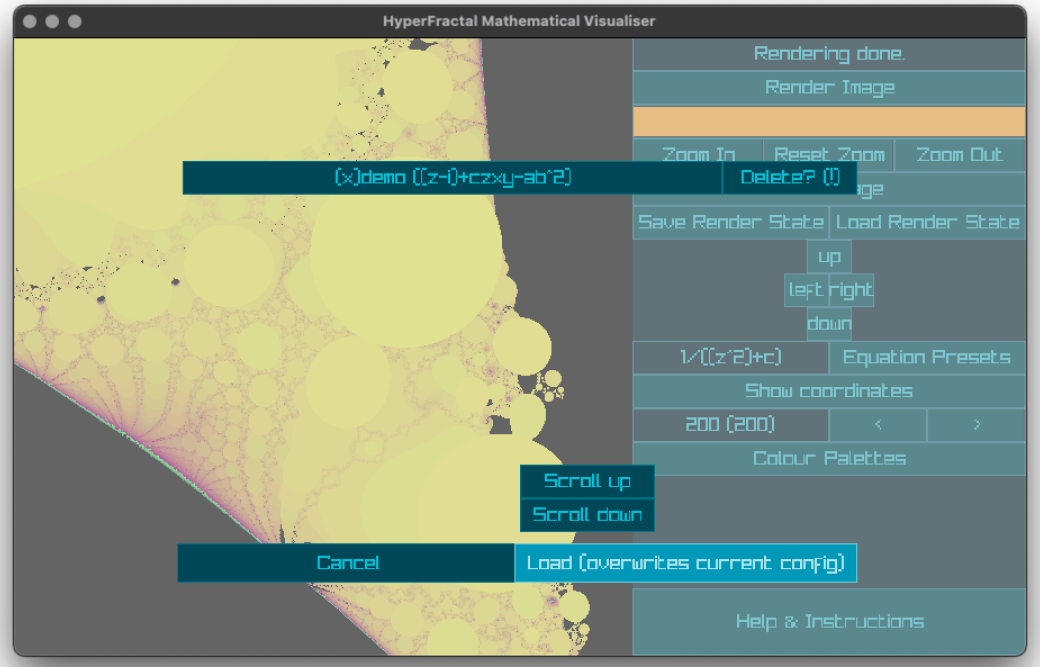
6. Pressing ‘Render Image’ produces a full-resolution render in under 5 seconds



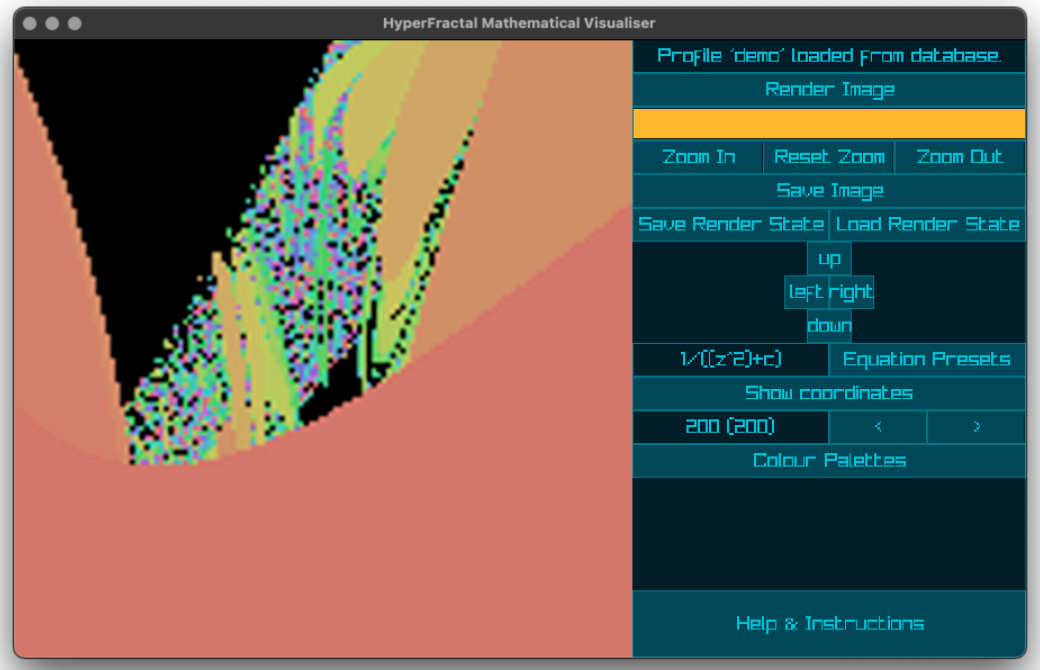
7. Clicking the ‘Colour Palettes’ button shows a list of available colour palettes



8. The colour palette has changed to be the one selected



9. Opening the configuration loading dialog in order to reload the ‘demo’ profile



10. Successfully reloading the ‘demo’ profile from the database

User Feedback

I gave my finished application to my maths teacher (my primary client) to use. After a brief introduction and them reading the instructions which come with the application, they very quickly got the hang of navigating using the interface, learning to use both the interface buttons and keyboard shortcuts with almost no direction from me. They found it easy and intuitive to switch between equation presets and colour palettes, and even started using the ability to save and load render states from the database. This clearly demonstrates that my interface design is easily usable for a teacher. It also demonstrated the interface's robustness when faced with a completely new user. For instance, the interface locking during a render feature was used at least once as my user tried to navigate the fractal during a render a few times. Thus, my program clearly satisfies high-level objective 1-08, as the interface is intuitive to my user.

One point that my maths teacher mentioned was how quickly images were rendered out. This is very good, as it indicates that the render times possible with the application are appropriate for an average user to tolerate (most users will not want to sit around waiting for a render to appear), and that my initial requirement on the render time for a fractal, under 30 seconds (high-level objective 1-07), was appropriate.

My maths teacher particularly liked the ability to select different colour schemes for renders, and quickly saw the usefulness of being able to save and load render configurations for demonstrations during lessons. As such, I'm very pleased with my solution as it clearly satisfies the requirements and expectations of my client.

Improvements and Further Evaluation

Overall, I'm very pleased with the completeness of this project. One thing which particularly surprised me was how efficient the rendering code is, despite it being relatively simple. As shown in my tests, standard renders using built-in equations happen in as little as 3 seconds, which I count as a major success.

However, there are a number of improvements I would have liked to have made. First, I would like to have expanded on the database management system, potentially adding the ability to export and import configuration profiles to share them between users. On top of this, I would have liked to be able to save a low-resolution preview of each profile along with the database, in order to make it easier for the user to find the profile they want. I would also have liked to add the ability to search the database by name or by other parameters (such as the equation being used). Here I would also have added the ability to create multiple user profiles and manage the database in a more central location (e.g. 'Program Files' on Windows or 'Application Support' under macOS).

Second, I would have liked to add support for saving PNGs. The application only saves images to a standard bitmap format, which is readable but does not show the colourful patterns shown in the interface. I was disappointed that I did not have time to implement this more complex method, but I felt that given the primary use of this application is in the GUI, the usage of this feature would be limited anyway.

Finally, during the development of my technical solution, I came up with an idea for a more efficient way of assigning pixels to computation threads, which was to assign them 'blocks' from the image, which they would gradually fill in. This would eliminate any blocking when threads were attempting to get new pixels from the image to be computed, potentially having a significant performance improvement. On top of this, I could have taken the time to write a GPU kernel in order to utilise host machine GPU capabilities. However, bearing in mind that I would have had to learn this from scratch, and that the majority of the target computers (school-issued teacher laptops) would have extremely limited, if not non-existent GPUs, I decided to stay with standard CPU based computation.

Appendices

Appendix 1 - Help Document

The following pages show the ‘Help & Instructions’ document which explains how to use the application.

Help & Instructions

Feeling lost? Below is an explanation of how to use the GUI mode of the application.

Rendering

When you start the application or make changes to parameters such as zoom, equation, centering, etc, you will be shown a low-resolution preview render in the viewport on the left side of the window.

By clicking on the 'Render Image' button in the control panel, the application will then generate a screen-resolution image in the viewport, during which time parameter controls are locked. Rendering progress is shown below the 'Render Image' button.

Parameters

Parameters determine the image which is output. These include the following:

- Zoom - allows you to look closer at particular areas by stretching the mathematical space on both axes - can be altered using the 'Zoom In', 'Reset Zoom', 'Zoom Out' buttons, or the '+' and '-' buttons on the keyboard
- Offset - allows you to move the viewport around the fractal by offsetting the mathematical space on one or more axes - can be altered using the 'up', 'down' 'left', 'right' buttons, or the arrow keys
- Equation - the iterative mathematical expression which is used to generate values for each pixel - clicking in the equation input box and typing allows you to alter this: **see dedicated help section**
- Iteration limit - limits the number of times the equation is iterated before the program assumes it does not tend to infinity - can be altered using the '<' and '>' buttons, or using the left and right square bracket keys (accelerate by holding Shift)
- Palette - can be selected from presets using the 'Colour Palettes' button

Saving and Loading Render States

The application allows you to load or save render states. A render state is a configuration profile containing all the rendering parameters used, including zoom

offsets, iteration limit, palette, etc.

In order to save a render state, click the 'Save Render State' button, click the input box in the middle of the screen, enter a descriptive name, and press 'Save'.

To load a render state, click the 'Load Render State' button, select an available saved profile from the list, and click 'Load'. This will overwrite current settings, so if you have a nice configuration currently showing, make sure to save it first before loading another. The loading dialog also gives you the option to delete saved profiles if you want, but be careful because this is permanent.

Equations

Fractals are generated by iteratively applying a formula to a complex number and counting the number of times it can be iterated before tending toward infinity. This count is then used to compute a colour for a given pixel.

Formulas (also referred to as expressions or equations) are in the format `Z = f(Z, C)`, where `Z` is referring to the iterated value, and `C` is constant. On the first iteration, `Z` and `C` are both initialised to be equal to a complex number, with components `a+bi`. `a` and `b` are the horizontal and vertical coordinates (respectively) of the pixel being computed, meaning each pixel has different initial conditions for computation.

The equation field in the application can handle brackets (`(...)`), indices (`^`), division (`/`), multiplication (`*`), addition (`+`) and subtraction (`-`), and processes them in that order. It supports the use of `z` and `c` as basic variables, as well as `x` and `y`, which behave as the real and imaginary part of `z`, and `a` and `b`, which represent the real and imaginary parts of `c`. It also supports the use of numerical (decimal) constants which can be in terms of `i`.

There is no limit to equation length or complexity, but more complex equations are likely to be more computationally expensive and increase render time significantly.

The application also contains a number of equation presets, which can be fun to explore and are good starting points if you want to come up with your own equation. Presets run much faster than custom entered equations due to them being hard-coded.

When writing your own equations, it is advised to replace `z^2` with `z*z`, as the latter evaluates much faster in a computational environment. The same can be done for any integer power of a function. In general, *you are advised to simplify your expression to a reasonable extent before entering it into the application, in order to reduce render time*.