Real-Time Fractal Rendering

Non-Examined Assessment
Optimised Fractal Rendering in

Toby Davis

Hampton School

March 16, 2023

Abstract

Write this bad boi when the thing is finished :)

Summary

1 Project Analysis **3**
1.1 A Brief Introduction to Fractals 3
1.2 Defining the Problem 4
1.3 The End User 6
1.4 Researching Existing Projects and User Requirements 7
1.5 The End User 9
1.6 Researching Existing Projects and User Requirements 10

2 Design Phase **12**
2.1 Third Party Libraries 12

1 Project Analysis

1.1 A Brief Introduction to Fractals

A fractal is "a curve or geometrical figure, each part of which has the same statistical character as the whole"[1].

Some fractals are defined by simple equations which exhibit chaotic behaviour. Arguably the most famous fractal, the Mandelbrot Set, is defined by the iterative equation, where $Z_0 = 0 + 0i$, and C is the initial value in the complex plane.

$$Z_{n+1} = Z_n^2 + C \quad : \quad Z_n, c \in \mathbb{C}$$

For a given point C to be in the Mandelbrot Set, the value of Z_n must remain bounded (i.e. not diverge to infinity) after the iterative series is repeated infinitely many times. This approach is used in most iterative fractal equations.

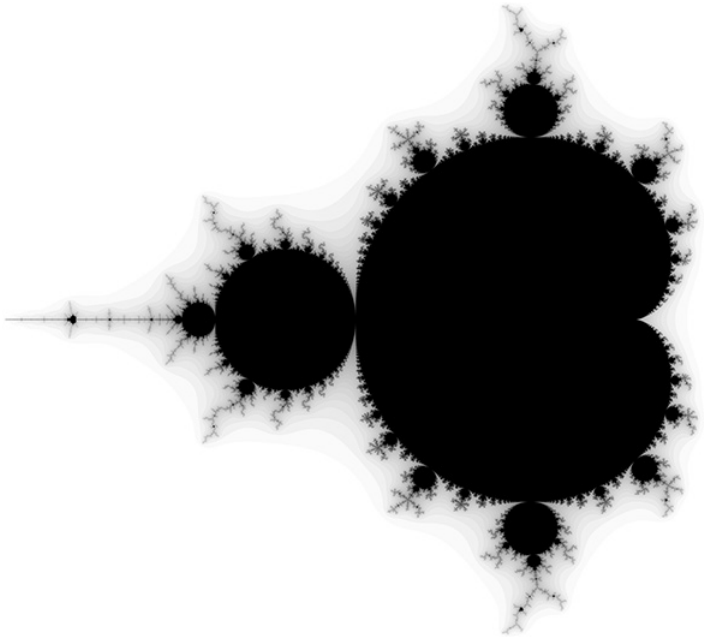


Figure 1: The standard Mandelbrot fractal[2]

Additionally, fractal variations can be created by changing the generating equation slightly. For example, changing the r in the Mandelbrot equation ($Z_{n+1} = Z_n^r + C$) yields the following fractals.

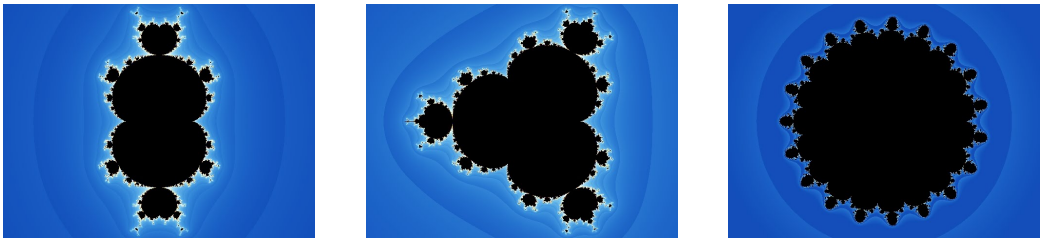


Figure 2: (left) $r = 3$, (centre) $r = 4$, (right) $r = 20$

Other famous fractals include the Sierpiński Triangle, the Julia Set, Hilbert Spirals, etc. All are defined either by infinitely-recursive self-similar patterns or repeated equations.

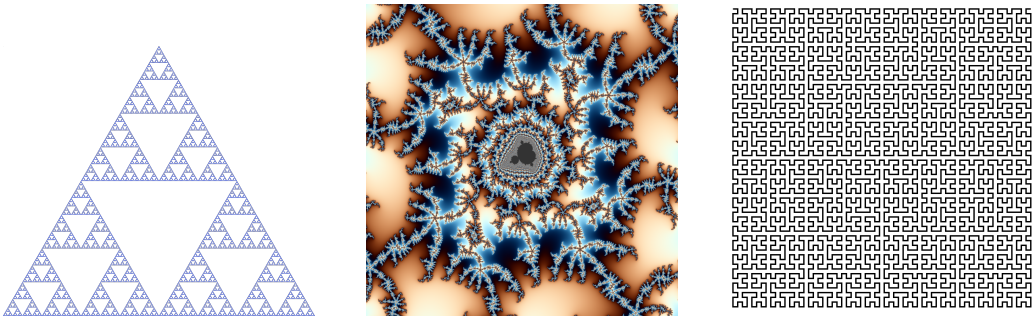


Figure 3: (left) Sierpiński-Triangle, (centre) Part of the Mandelbrot Set, (right) Hilbert Curve

1.2 Defining the Problem

Fractals have been the subject of much debate and curiosity throughout history, but, due to the computational requirements of generating them, research into them was very limited until the rise of the electronic computer.

The newfound levels of processing power allowed increasingly detailed images to be generated, and mathematicians were able to better understand the underlying equations and seemingly chaotic nature of fractals.

With the power of modern computers, it is possible to render some fractals in real-time and explore them to great depths, though there are still technical, physical and monetary hurdles to clear.

Many fractal rendering program exist online, however the majority of these are incomplete, inefficient applications that are not designed for high-performance and high zoom factors. While there are high-quality applications out there, the best ones are often quite expensive, making them inaccessible to the majority of potential-users. For example, some extremely advanced software costs almost £80[3].

Technological Limitations The further you zoom into a fractal, smaller the numbers you have to deal with. At lower zoom levels, this doesn't pose much of an issue, as 64 bit or even 32 bit floating point numbers often have the required precision to accurately render an image. At higher zoom levels, however, the precision of the numbers used starts to affect the image quality.

When zoomed in far enough, floating point rounding errors start to cause certain pixel positions to merge into one, causing unattractive "blocky" patterns in the image. Eventually, these blocks will consume the entire image and no more detail can be seen.

To get around this, it is possible to use high-precision floating point data types, though, since these are processed in software, not hardware, they are orders of magnitude slower than normal number types, and this can make the rendering process impractically slow.

Certain approaches can be taken to optimise the performance of high-performance number types. For example, it can be proven that Z_n will diverge to infinity if $|Z_n| > 2$ at any point. Additionally, advanced algorithms can mix fixed and multi-precision arithmetic to decrease the number of arithmetic operations that take place in software.

Program Limitations Many fractal renderers do just that; render fractals. They don't support any kind of render export and do not allow for saving or reloading render configurations.

Some programs have methods to save the rendered fractals as image files, but often have limiting export settings and don't support many resolutions. A wider range of programs allow the current position and zoom level, among other information, to be exported in a file. This allows the configuration to be reloaded and interesting locations in fractals to be shared easily.

Precision vs Performance To increase rendering performance, most implementations of fractal renderers use strictly 32- or 64-bit floating point numbers. Since these data types are operated on directly by the hardware, they are extremely efficient. Unfortunately, 64-bit floating point values can only represent around 15 decimal places accurately, so zooming in far enough will exceed this precision and cause visual glitches.

To circumvent this issue, it is possible to use multi-precision floating point types that are capable of representing hundreds, thousands or even millions of bits, allowing for near-infinite zooms. These numbers, however, are implemented in software and are many orders of magnitude slower than standard floating point types.

For sufficiently optimised programs, it is possible, to a point, to use multi-precision floating point arithmetic, though the performance will be very poor.

Furthermore, some areas of different fractals require an immensely large number of iterations before a reasonable amount of detail can be obtained. A direct implication of this is that potentially millions of calculations must be done to determine the colour for a single pixel.

The two main issues above become even more extreme when combined with the goal of near-infinite zooming. Due to the nature of many fractals, the number of iterations required to get high levels of detail in areas close to the border of the fractal increases with zoom. Additionally, deeper zooms require higher precision numbers to accurately represent all the points. Combine these, and the result is an extremely slow, inefficient program which is not usable, in many cases.

1.3 The End User

A fast, fully-featured fractal viewer could be used by a wide range of people for a variety of different purposes.

Firstly, maths teachers could use the program to assist in their lessons, as well as providing students with an interactive resource to help with homework and to further their understanding.

Secondly, if the renderer is fast and precise, even at extremely high zoom levels, then it could be used by people researching fractals and examining their properties.

Finally, such a program could also be used by anyone who enjoys looking at fractals, which, most likely, provides a larger audience than the other users

combined.

Combining these different user-bases, an efficient, intuitive fractal renderer could be used by a significant number of people for a broad spectrum of different purposes.

User Requirements

Unfortunately, each of the identified user groups will have a different set of specific requirements to be met. These will be covered later in the project, but it is likely that only a limited set of features can realistically be implemented.

1.4 Researching Existing Projects and User Requirements

There are many fractal viewers freely available online already, though almost all of them suffer from at least one very significant flaw:

1. Lack of precision
2. Inadequate performance
3. Insufficient customisability
4. Limited file export options
5. No ability to save configuration to a file

Lack of precision

To increase performance, many simpler implementations use strictly 32- or 64-bit floating point values. Additionally, some implementations greatly reduce the number of samples run per pixel, meaning the fractal is lacking critical detail in some areas – particularly those closest to the border of the fractal.

While it is possible to increase the maximum number of iterations run per pixel, moving from 64-bit (machine word) to multi-precision (software) floating point incurs a *massive* performance hit, making it impractical in almost all cases.

Inadequate performance

Many naive fractal renderers do not implement the necessary optimisations for the program to run at real-time with a native resolution. This links closely with the lack of precision, since the two are extremely closely related.

Insufficient customisability

When rendering and exploring fractals there are a lot of settings that can change how it appears. For example, you can change the maximum number of iterations to allow, or the settings/algorithm for colouring the fractal.

Unfortunately, the vast majority of programs limit the customization of the fractal to the gradient used to colour it, not allowing the user to change any other settings.

Being able to change the most fundamental settings of the renderer allows for anyone using the software to gain a more intuitive understanding of both the fractal itself and the program.

Limited file export options

A surprising number of fractal rendering programs do not allow the final product to be saved to a file, which is a nice feature to have if you want to examine them further in more software, for example.

The ability to save configurations

Often, when exploring fractals, you come across areas of particular interest and would like to be able to return to that exact place in the future.

Very few programs actually implement this feature, and it is one which could be very useful in some circumstances.

1.5 The End User

A fast, fully-featured fractal viewer could be used by a wide range of people for a variety of different purposes.

Firstly, maths teachers could use the program to assist in their lessons, as well as providing students with an interactive resource to help with homework and to further their understanding.

Secondly, if the renderer is fast and precise, even at extremely high zoom levels, then it could be used by people researching fractals and examining

their properties.

Finally, such a program could also be used by anyone who enjoys looking at fractals, which, most likely, provides a larger audience than the other users combined.

Combining these different user-bases, an efficient, intuitive fractal renderer could be used by a significant number of people for a broad spectrum of different purposes.

User Requirements

Unfortunately, each of the identified user groups will have a different set of specific requirements to be met. These will be covered later in the project, but it is likely that only a limited set of features can realistically be implemented.

1.6 Researching Existing Projects and User Requirements

Look at online renderers (including Google's one??) Look at offline renderers.

Pick out the best features

Talk to Josh Talk to teachers Talk to family

Identify the overarching requirements Identify things that would be nice to have, but not necessarily required

</> Test Box

```
#include<iostream>

int main() {
  std  cout << "Hello, World\n"
  return 0;
}
```

Details Gathered – TEMPORARY

Online Renderer

1. https://math.hws.edu/eck/js/mandelbrot/MB.html
2. Rectangle zoom box is AWFUL
3. Settings are useful

   Adjustable resolution

   Number of threads

   Palette editor

   Configuration Export

   Offers multiprecision floating point

XaoS Renderer

1. https://fractalfoundation.org/resources/fractal-software/
2. Method of moving around is quite clunky
3. Almost every setting can be changed, which is good
4. Multiple fractals supported
5. Different colouring and shading options
6. Some VERY interesting algorithms at play for performance boosts

   Some form of data retention between frames

   Progressive rendering

   Looks like a 128-bit float but no true multiprecision floating point

   Multiple approximations followed by progressive refining of the fractal

2 Design Phase

2.1 Third Party Libraries

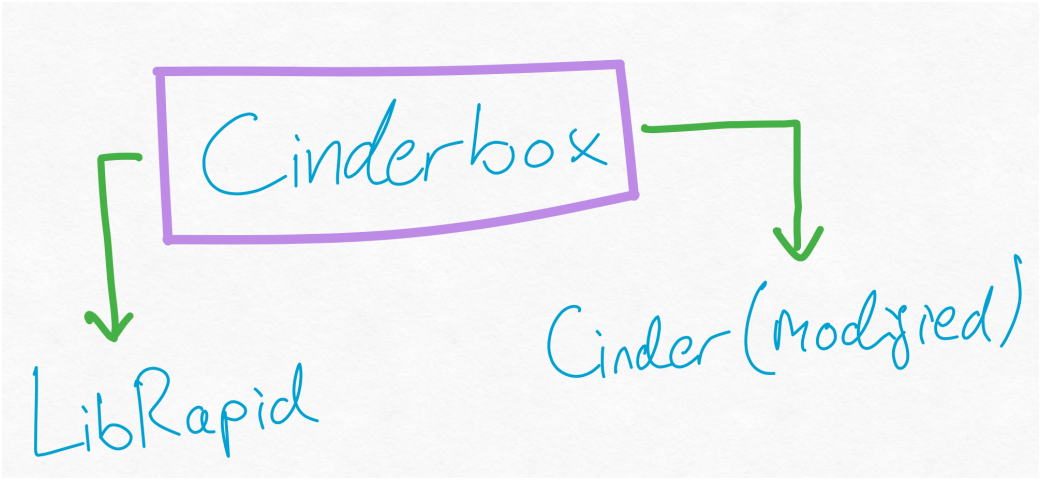
Cinder [4] *Cinder* is a free, open source graphics engine for C++. It provides a simple way to access OpenGL, ImGui and other tools, such as image loading and saving, optimised rendering in 2D and 3D, and more. I am using *Cinder* for this project instead of doing all the graphics processing with raw OpenGL because it dramatically simplifies the code, and reduces the scope for hard-to-fix bugs.

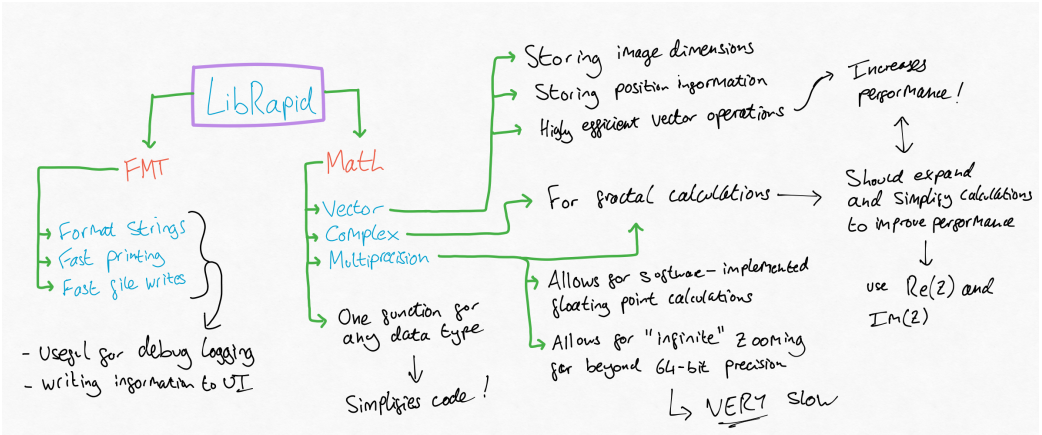
Furthermore, I am using a custom fork of the library with my own changes applied to the code, fixing some issues with it and adding some more functionality.

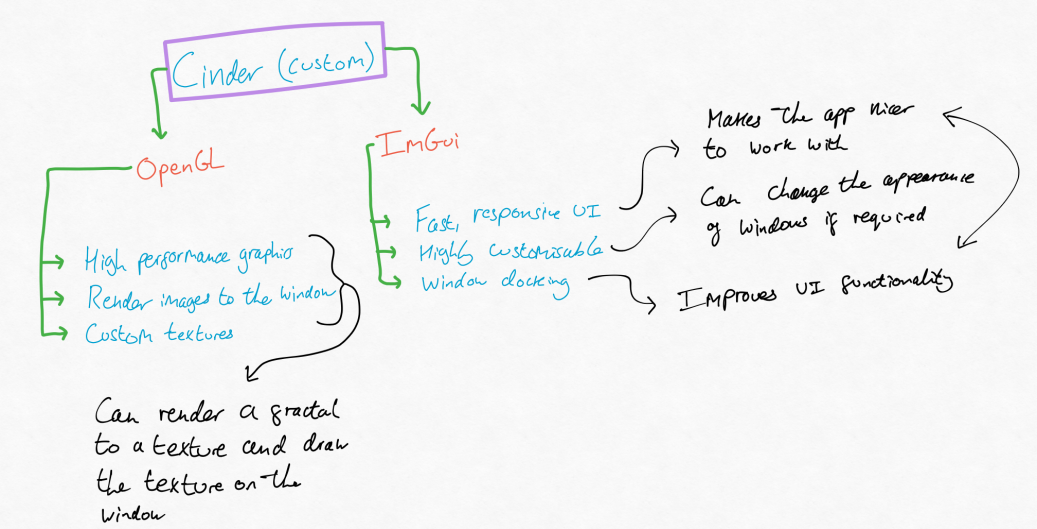
LibRapid [5] *LibRapid* is a high-performance library for mathematical applications, including optimised vector classes, complex number types and general mathematical functions. The most useful feature for this project, however, is its support for MPIR and MPFR, which are multiprecision implementations.

Being able to incorporate an efficient multiprecision implementation into the project could allow for "infinite" fractal zooms, since the software would no longer be constrained by floating-point limitations.

Cinderbox Both of the afore mentioned libraries are packaged with *Cinderbox* for simple integration into *CMake* projects.


Cinderbox
LibRapid
Cinder (modified)


LibRapid
FMT
Format strings
Fast printing
Fast file writes
- Useful for debug logging
- writing information to UI
Math
Vector
Complex
Multiprecision
One function for any data type
Simplifies code!
Storing image dimensions
Storing position information
Highly efficient vector operations
Increases performance!
Should expand and simplify calculations to improve performance
use Re(z) and Im(z)
For fractal calculations
Allows for software-implemented floating point calculations
Allows for "infinite" zooming far beyond 64-bit precision
VERY slow



References

[1] Catherine Soanes and Sara Hawker. *Compact Oxford English Dictionary of current English*. Oxford University Press, 2013.

[2] Paul Bourke, Nov 2002. URL http://paulbourke.net/fractals/mandelbrot/.

[3] Frederik Slijkerman. Ultra fractal: Advanced fractal software for windows and macos, Jul 2022. URL https://www.ultrafractal.com/.

[4] Cinder Developers and Toby Davis. Pencilcaseman/cinderbox: A modified version of cinder with extra libraries for improved functionality. URL https://github.com/Pencilcaseman/cinderbox.

[5] Toby Davis. LibRapid: Optimised Mathematics for C++, 1 2023. URL https://github.com/LibRapid/librapid.