OCR Computer Science H446

NEA Submission May 2023

Ace Harvey

Centre No 20153

Candidate No 4

## Project Overview

My project is a piece of software designed to visualise complicated mathematical systems, such as the Lorenz attractor, and other similarly complex “chaotic” systems.

* . The project should visualise these by rendering high-resolution high-fidelity images which trace a point in the system as time progresses.
* The project could also produce videos of a system as it evolves over time, rather than a single image.
* The project should prioritise quality of image, as well as efficiency.
* The project should also be highly configurable, such as changing scale, colours and rendering algorithms easily.
* Users should be able to enter their own custom governing equations for systems
* Users should be able to save and load configuration files so that they can easily recreate images with slight changes if needed.
* The project should also have a graphical user interface, and could include a small preview window.

Contents

[Project Overview 1](#_Toc115856426)

[Analysis – Problem Identification 4](#_Toc115856427)

[Analysis – Stakeholders 5](#_Toc115856428)

[Identifying Stakeholders 5](#_Toc115856429)

[Survey Questions 6](#_Toc115856430)

[Response: Stakeholder Mark 6](#_Toc115856431)

[Response: Stakeholder Isabelle 6](#_Toc115856432)

[Analysis of responses 6](#_Toc115856433)

[Analysis – Computational Methods 7](#_Toc115856434)

[Abstraction 7](#_Toc115856435)

[Decomposition 7](#_Toc115856436)

[Logical nature 7](#_Toc115856437)

[Procedural nature 7](#_Toc115856438)

[Input and output 7](#_Toc115856439)

[Saving configurations and outputs 7](#_Toc115856440)

[Why is this project suited to a computational approach? 8](#_Toc115856441)

[Analysis – Existing Solutions 9](#_Toc115856442)

[*Visions of Chaos* 9](#_Toc115856443)

[Summary 9](#_Toc115856444)

[Features 9](#_Toc115856445)

[Technological Analysis 9](#_Toc115856446)

[*Chaoscope* 10](#_Toc115856447)

[Summary 10](#_Toc115856448)

[Features 10](#_Toc115856449)

[Technological Analysis 10](#_Toc115856450)

[*glChAoS.P* 11](#_Toc115856451)

[Summary 11](#_Toc115856452)

[Features 11](#_Toc115856453)

[Technological Analysis 11](#_Toc115856454)

[Analysis - Features 12](#_Toc115856455)

[Analysis - Limitations 13](#_Toc115856456)

[Analysis - Requirements 14](#_Toc115856457)

[Analysis – Success Criteria 15](#_Toc115856458)

[Design - Decomposition 16](#_Toc115856459)

[Design – Structure 17](#_Toc115856460)

[Design – Algorithms 18](#_Toc115856461)

[Design – Features 19](#_Toc115856462)

[Design – Key Jawns 20](#_Toc115856463)

[Design – Testing Data 21](#_Toc115856464)

[Design – Further Data 22](#_Toc115856465)

# Analysis – Problem Identification

TODO: THIS

# Analysis – Stakeholders

## Identifying Stakeholders

The stakeholders for the project are either people interested in the mathematics of chaos, looking to visualise and/or analyse chaotic maps, or people that want to use the software for the soothing visuals often created by similar software. I would like the software to meet the requirements of both stakeholders.

The software should have inbuilt methods for mathematical analysis, such as searching through parameters and XY phase-space to find attractors and repulsors and potentially identify their nature. The software should also have the ability to input custom systems of equations so that new attractors can be discovered and analysed. As such, the software would be useful to the first kind of stakeholder as they would be able to use it for mathematical visualisation and analysis. Also, batch and video rendering could be useful to analyse how a system evolves as parameters change or over time.

The software should also be able to render maps with custom colouring and high-quality rendering, potentially in real-time so that users looking to use the software for the soothing nature of the generated images and videos It should also include built-in examples and an easy to understand interface without the need of complicated mathematics to use at basic level.

I will survey both types of stakeholder to gain more of an insight into what those stakeholders individually want out of the software, and how it can be more suited to their needs.

## Survey Questions

1. What would you use the software described for, and why?
2. What existing solutions have you heard of / used in the past?
3. If you have used existing solutions, what did you like/dislike about those solutions?
4. How often have you used said existing solutions?
5. Are there any specific features you would like to be implemented?

## Response: Stakeholder Mark

1. **What would you use the software described for, and why?**
   1. I would use the software for visualisation purposes, as existing software doesn’t allow me to visualise systems that aren’t pre-loaded into the software. I would also use it to search through and analyse the phase space, looking for attractors to classify.
2. **What existing solutions have you heard of / used in the past?**
   1. I have used glChAoS.P and Chaoscope.
3. **If you have used existing solutions, what did you like/dislike about those solutions?**
   1. I really liked the user interface on glChAoS.P but it didn’t allow me to add my own systems or render out videos as a parameter is changed. I also felt the same about Chaoscope, but that had a lot of other features I liked such as batch rendering, and the search window which helped find interesting parameter combinations and points in phase space.
4. **How often have you used said existing solutions?**
   1. I did not often use the existing solutions because of the aforementioned issues.
5. **Are there any specific features you would like to be implemented?**
   1. Aside from what I mentioned earlier, I’d like to be able be able to use the program on Linux because that’s my main operating system.

## Response: Stakeholder Isabelle

1. **What would you use the software described for, and why?**
   1. I would use the software in a therapeutic way to make cool visuals in order to relax.
2. **What existing solutions have you heard of / used in the past?**
   1. I haven’t heard of anything exactly like it, but the closest thing I can think of is like blender or Photoshop, something related to 3d modelling.
3. **If you have used existing solutions, what did you like/dislike about those solutions?**
   1. N/A
4. **How often have you used said existing solutions?**
   1. N/A
5. **Are there any specific features you would like to be implemented?**
   1. I would like to be able to change the color scheme. I would also like the interface to be easy to use, maybe with a little information button telling me what each slider does.

## Analysis of responses

From these responses, I deduced that the creation of the project is justified as Mark had issues with existing solutions and Isabelle was unaware of similar existing solutions. The project meets the needs of both Mark and Isabelle I also deduced that the following features will be important to implement:

* Render a video as a parameter is gradually changed
* Batch rendering
* Search window
* Linux Compatibility
* Ability to change color scheme
* Tooltips or similar on the user interface

# Analysis – Computational Methods

## Abstraction

The project should take a config file as an input, or take input from a UI to change said config file, then output a picture or video in the format selected by the user. However, processing of data into images is a complicated problem, and one that has been solved by many before me. As such, for some elements of output processing (mostly pertaining to encoding raw pixels into filetypes) I will be using existing libraries such as PIL (Python Image Library) and openCV for video encoding. This usage of existing libraries is abstraction, a computational method.

Furthermore, I could use abstraction to hide away the complicated workings of the project, only showing the user the parameters that affect the look of the final image rather than all of the parameters for image rendering which, when adjusted, could cause unexpected results.

## Decomposition

The project decomposes into various tasks easily, such as dealing with the mathematical expressions, the image processing and rendering, etc. By using decomposition in this way, I can break down the problem into less complicated problems and solve those individually, which would be easier than tackling the whole problem at once.

## Logical nature

The project requires the use of logic in many different aspects – such as validating mathematical expressions, evaluating said mathematical expressions and validating configuration files.

## Procedural nature

The project is also highly suited to procedural programming techniques, as the program can be decomposed into smaller programs and then recombined at the end. Doing so would also allow me to reuse different parts of the project in other parts of the project, or potentially in future projects. Building smaller subprograms and combining them allows for a smoother workflow and also allows me to isolate problems or bottlenecks easily.

## Input and output

Presenting output is a problem that leads itself to the use of computational methods as the simplest way to present an output diagram is through an image or video on a computer screen – attempting to do otherwise would be complicated, requiring some form of physical image creation which could be expensive due to the need for physical materials and would be inconvenient if the user made a mistake in input parameters and had to re-create the image again, costing double. As such, I believe a computational approach is best to present the output.

Furthermore, getting input to the project would be difficult without a computational approach as I would like to present the parameters in an easy to use and understand way. This leads itself to using a computational approach as I could easily create a graphical or command line interface that labels and explains various parameters and allows the easy entry of numbers or equations through use of a mouse and/or keyboard.

## Saving configurations and outputs

In addition, the project should be able to save and load inputs via the use of configuration files, which is easy to implement via computational methods such as file handling (such as pythons `open` syntax), but would be difficult to do without a computer. Using a computer avoids the need to either remember the desired configuration, or have a physical way of storing or inputting parameters.

Also, the project should be able to save its outputs, which again would be easy to do via file handling on a computer. Other approaches could be inconvenient for the user, for example, saving images using a computer would allow the user to then analyse or process them further using other computer software, in a way that would be difficult if not impossible should a computational approach not be used for saving outputs.

## Why is this project suited to a computational approach?

As I have detailed, there are many reasons I believe that make this problem solvable via use of a computational approach; input, output and processing would all be challenging to implement without use of computational methods, and as such I believe that the whole project should be done via a computational approach. Furthermore, the usage of libraries to process raw image and video data would not be possible without the use of a computer, and as such the project is highly suited to a computational approach.

# Analysis – Existing Solutions

## *Visions of Chaos*

### Summary

*Visions of Chaos* is a windows program made by *Softology* focused on simulating various mathematical models. The software is available for all versions of windows, and focuses on being an all-in-one tool for rendering chaotic models. The software is kept up-to-date but the attractors module does not get frequent updates. The software accomplishes some of the goals of the project, but is more focused on creating soothing images rather than providing tools for mathematical analysis or searching through parameter combinations.

### Features

The software, however, has features the stakeholders expressed interest in:

* Need to actually use this software at some point

### Technological Analysis

A

## *Chaoscope*

### Summary

*Chaoscope* is a windows program made by *Nicolas Desprez* with a linux-compatible command-line version. The software focuses on rendering 3d attractors, rather than 2d models, but is still a very powerful piece of software. The software was last updated on the 31st Oct 2010, so is relatively outdated now. The software does not have support for user defined attractors, but instead uses a group of predefined attractors that potentially have undiscovered parameter combinations that lead to mathematically interesting behaviour.

### Features

I presented the software to my two stakeholders. The software has the following features that the stakeholders expressed interest in, or liked:

* Command line batch rendering
* Project files that save parameters, viewports, etc
* Search window for searching for parameter combinations that lead to interesting behaviour
  + Graph based method, where parameters are mapped to an XY plane
  + Automated method that randomizes parameters
  + Randomness and parameter exclusion to help automate searching
  + Sliders to change parameters slightly
* Level parameter for root-based attractors (i.e. Julia attractor)
* Preview window.

And they disliked the following:

### Technological Analysis

A

## *glChAoS.P*

### Summary

*glChAoS.P* is a piece of software designed to utilise openGL to visualise and explore various 3d fractals. It runs in Linux, Windows, MacOS and webGL. The project was created and is maintained by *Michele Morrone*. The software is not longer actively maintained, with the last update in 2020. The software had a focus on fast rendering and having a range of preset systems, ranging from attractors to hypercomplex fractals.

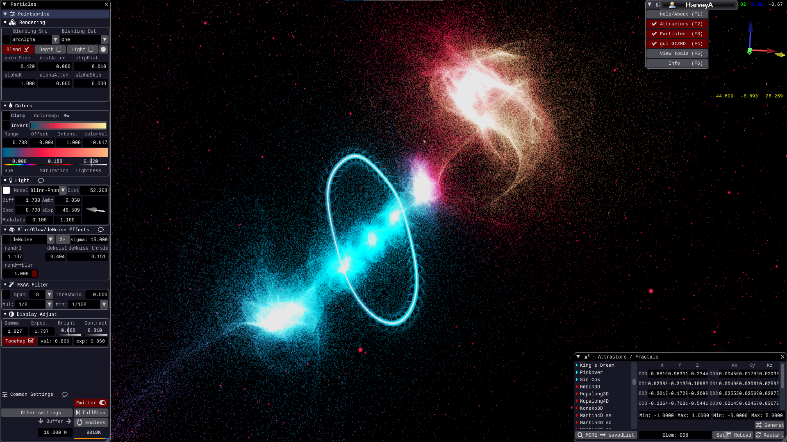


Figure 1: wglChAoS.P rendering a fractal. On the left is shown the rendering settings

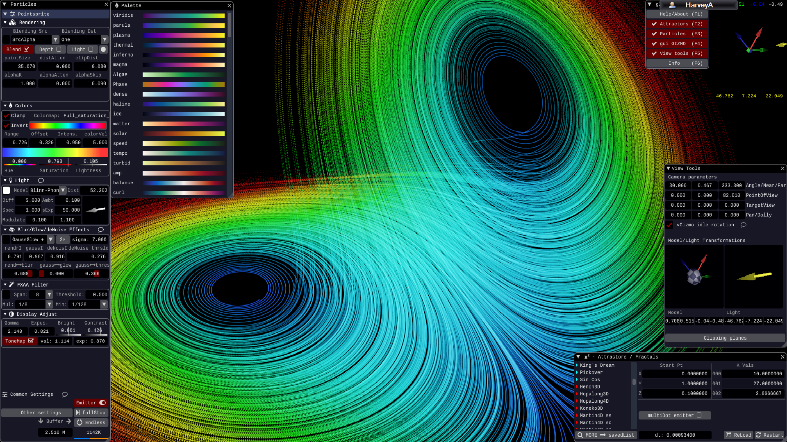


Figure 2: wglChAoS.P rendering the Lorenz attractor. Rendering, colour, and viewport settings are shown.

### Features

I presented the software to my two stakeholders. The software has the following features that the stakeholders expressed interest in, or liked:

* The ability to change, edit and save colormaps makes fine tuning the look and feel of a render easier.
* Compatibility with multiple operating systems, so the end user can use the software on linux
* Easy to use camera orbiting using the mouse is very intuitive.
* Multiple emitter modes, allowing the user to choose the particle size and the overall look.
* De-noise and glow features allow for better-looking images
* FXAA Anti-Aliasing allows for higher fidelity images
* Easy to use parameter UI with the ability to type or drag parameters
* Window-based UI with a preview in the background makes adjusting settings very easy

And they disliked the following:

* Inability to input custom systems of equations.
* No support for 2d systems, only 3d systems.
* Does not utilize the GPU as well as it could.
* Lighting system seems unnecessary.

### Technological Analysis

The software uses openGL for rendering, and is programmed in C++. This also allows the program to use webGL and also can utilise the speed of OpenGL. The fractals and attractors are rendered using a dot emitter that emits particles that lie along the path of a point as it evolves over time. The colormap represents when in time the point would have evolved. This approach is effective because as the rendering continues the picture gets more and more accurate. The billboard mode also allows for the illusion of solid surfaces from solid spheres around the emitted points. This rendering method, however, strains the GPU as millions of points must be rendered simultaneously every frame, with more being added every second as the emitter functions. This approach might not be effective in Python due to its slow speed, but further testing will be required.

# Analysis - Features

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | | | **Source** | **Description** |
| Interface and user experience | Graphical User Interface (GUI) | Sliders to adjust values | Me | Sliders can be used to accurately and easily adjust values. |
| Text boxes for data entry | Text boxes can be used for entry of equations and precise values. |
| Radio buttons for Boolean choices | Radio buttons can be used for easily toggling Boolean choices. |
| Camera rotating and panning via mouse and keyboard | Existing solutions | Click-and-drag can be used to move the viewport around in space and the Q and E keys can be used to rotate the viewport. The scroll wheel can be used to zoom in/out. |
| Tooltips | Isabelle | Tooltips describe parts of the GUI. |
| Preview Window | Existing solutions | A preview window is present in the GUI. |
| Command Line Interface (CLI) | Command line operation of the program | The program can be operated from the command line. |
| Command line batch rendering | Mark | Multiple images can be rendered at once using the command line |
| Rendering | Rendering Stills | Changing Colormaps | Isabelle | The colormap of the image can be changed by the user. |
| Changing Viewport | Existing solutions | The viewport can be moved, zoomed or rotated by the user. |
| Rendering Videos | Rendering videos with time as the time dimension | Me | The user can choose to watch as a system evolves by rendering multiple images at different time values and stitching them together. |
| Rendering videos with parameters as the time dimension | Instead of at different time values, the rendering can be done using different parameter values. |
| Render Previews | Render still previews | Existing solutions | The GUI has a preview window that renders low-quality fast preview stills. |
| Render video previews | The GUI has a preview window that renders low-quality fast preview gifs. |
| High quality/fidelity | Super sampling | Me | The image is supersampled via a supersampling algorithm. |
| Anti-aliasing | The image is anti-aliased in some way. |
| High-res | The image can be up to 4k resolution, |
| Saving and loading | Saving and loading project files | Saving | Existing solutions | The user can save and load a whole project |
| Loading |
| Saving and loading config files | Saving | The user can save and load just config settings. |
| Loading |
| Mathematical / Analysis | Range of built-in mathematical functions | Trigonometric | Me | Trig functions can be used in an equation. |
| Exponential | Exponential functions can be used in an equation. |
| Matrices/vectors | Matrices/vectors can be used in an equation. |
| Piecewise | Piecewise functions can be used in an equation. |
| Hyperbolic | Hyperbolic functions can be used in an equation |
| Search function | Parameter searching | Me | Search through parameter space or phase space via a search window with customisable axis. |
| Space searching | Mark |
| Entering custom equations | The user can enter their own custom system equations | The user can, using the above functions) define equations for and and render images for those systems. |

Key: Orange text represents optional features.

# Analysis - Limitations

## Dimensionality

The software will have some limitations. The software will only deal with 2-dimensional systems as 3d rendering is outside of the scope of the project as it would be very difficult to develop and implement. As such, certain “classic” examples of chaos will not be able to be rendered by the program without losing a dimension.

## Efficiency

Another limitation is the slow speed at which python, a high level programming language, runs at compared to lower level languages such as Rust or C++. The rendering of higher quality images will take much longer and the real-time rendering of the preview window may be very slow. This could be remedied by using a lower level, faster language however I am not comfortable enough in those languages to develop a solution using them.

## Mathematical

The maths surrounding chaos theory

# Analysis - Requirements

# Analysis – Success Criteria

# Design - Decomposition

# Design – Structure

# Design – Algorithms

# Design – Features

# Design – Key Jawns

# Design – Testing Data

# Design – Further Data