OCR Computer Science H446 NEA Submission May 2023

Ace Harvey

Centre No 20153

Candidate No idk

Project Overview

My project is a piece of software designed to visualise complicated mathematical systems, such as the Lorenz attractor, and other similarly chaotic and interesting systems. It will take an input "project file" which contains information such as a system of differential equations, viewport settings and color settings and produce output images that show the phase space of that system. These project files can be generated by the program through use of a GUI made in Python.

Contents

Analysis – Problem Identification

Currently, no software exists to visualise any given system of equations relevant phase space. There exists much software made to visual phase spaces of common, known chaotic or interesting systems, however these pieces of software mostly only function in 3D and do not have support for entering custom, user defined equations. Entering custom equations if important as it may help to discover more strange or mathematically interesting attractors and help further the field of chaos theory and gain a better understanding of chaos and phase spaces.

A computer would need standard periphery to visualise these phase spaces, and this is suited to a computational approach as interacting with and creating images is much easier using a computational approach.

Such software is needed as the field of chaos theory is hard-to-access and is also very new in mathematics. Allowing people to experience this though interesting or soothing visuals could potentially help the field grow and progress, as well as help existing researchers visualise new chaotic attractors without needed knowledge of complicating programming and rendering. Furthermore, the interesting visuals can be used for their calming and smoothing effects by end users who are less interested in the mathematics.

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 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?
    a. 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?
    a. I have used glChAoS.P and Chaoscope.
3. If you have used existing solutions, what did you like/dislike about those solutions?
    a. 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?
    a. 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?
    a. 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?
    a. 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?
    a. 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?
    a. N/A
4. How often have you used said existing solutions?
    a. N/A
5. Are there any specific features you would like to be implemented?
    a. 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?

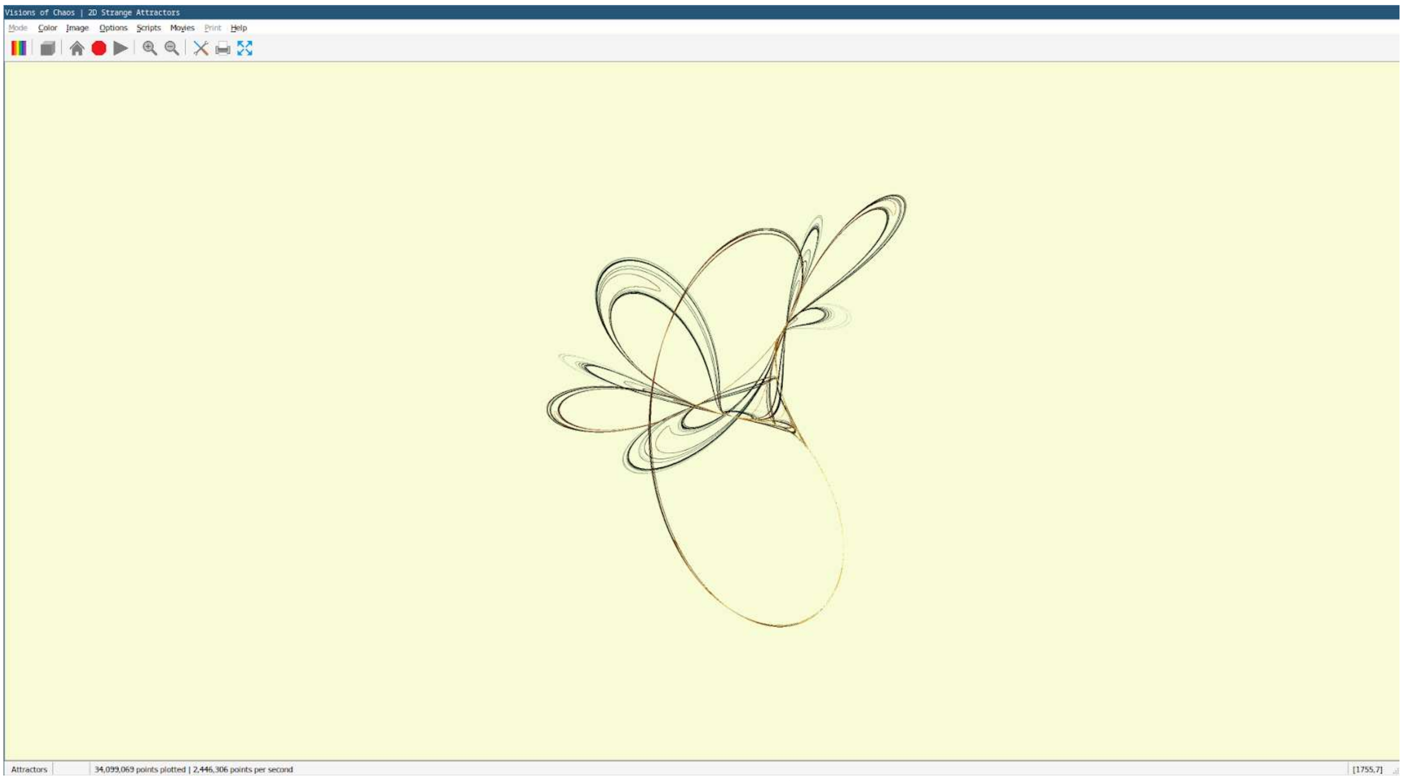
As I have detailed, there are many reasons I believe that make this problem solvable via use of a computational approach; input, output and processing would all be challenging to implement without use of computational methods, and as such I believe that the whole project should be done via a computational 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.



Visions of Chaos | 2D Strange Attractors
Mode Color Image Options Scripts Movies Print Help

Attractors | 34,099,069 points plotted | 2,446,306 points per second | [1755,7]

Figure 5: The program running in Wine

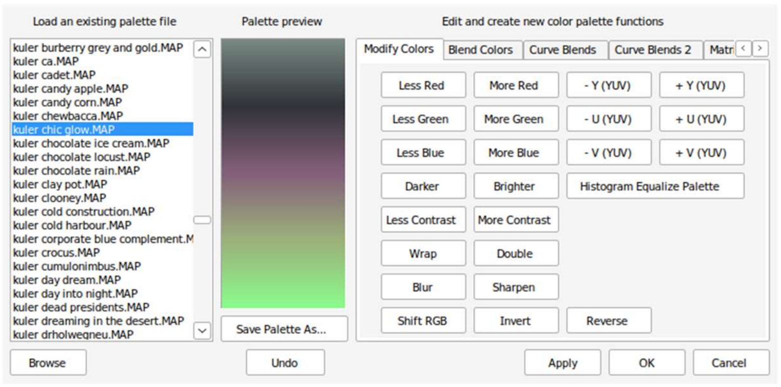
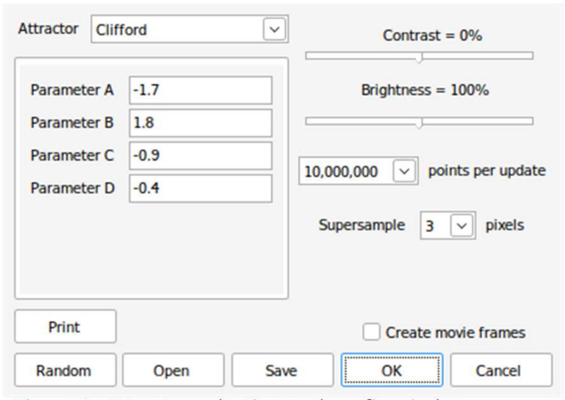


Figure 4: The color gradient editor



Attractor: Clifford

Parameter A: -1.7
Parameter B: 1.8
Parameter C: -0.9
Parameter D: -0.4

Contrast = 0%
Brightness = 100%

10,000,000 points per update
Supersample 3 pixels

Figure 3: Attractor selection and config window

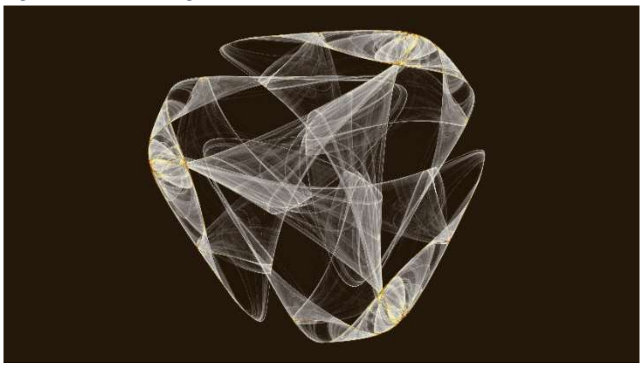


Figure 2: Example image generated by Visions of Chaos

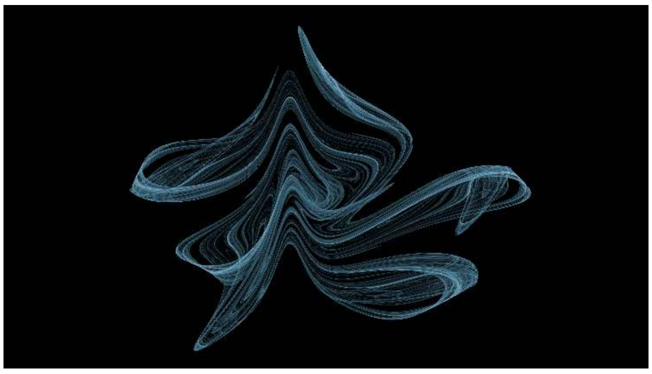


Figure 1: Example image generated by Visions of Chaos

Features

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

- A range of built in colormaps makes customization easy
- Powerful colormap editor that makes creating complex colormaps easier.
- The viewport settings are intuitive and easy to use, utilising the mouse rather than number or sliders.
- Creating videos is simple and fast via use of the move maker dialogue.
- Supersampling is effective at increasing detail and fidelity.
- Program runs well on low-end machines.
- Images can be exported in a range of ways, including directly to the wallpaper in windows.

The stakeholders, however, had some issues.

- Changing parameters is awkward and clicking through menus to do so is time-consuming.
- Some settings are not explained well in tooltips.
- Program can freeze up when trying to stop rendering.
- Cannot enter custom equation.
- Linux support under wine is slow and difficult to use.

Technological Analysis

This software first generates a list of points based on the formulas for the attractor, with some xstart and ystart value where x and y first start. Then, a list of millions of points is created, ordered by each iteration of the xnew and ynew formula. Then, for 3d models, cylinders are drawn between each consecutive point – i.e. in ordered list [A,B,C] the points A and B, and B and C will be joined. For 2d attractors, points are simply plotted on a lattice at least 3x the width and height of the final image, then a supersampling algorithm is applied to the larger grid to downscale the resulting image by 3x. This leads to a higher fidelity, antialiased image with less "jaggies" and with more detail. Note that the larger lattice is clamped between 0.5 and -0.5 in both X and Y dimensions, and all points scaled into this window so that the window always contains all points. This means there is no need for a viewport or viewport settings, but could lead to some attractors being squished or stretched in one or both dimensions. Then, the coloring of each cylinder in 3d or point in 2d is based on the initial co-ordinates of the point or start of the cylinder by converting each co-ordinate axis to a gradient colormap, usually XYZ to RGB but for 2d maps this could be XY to any gradient map. Then, the black background color is replaced with the specified background color. Custom attractors are not explicitly supported, however the author takes suggestions on attractors to add, and adding attractors is possible but difficult even with some knowledge of programming, via the built-in code editor.

The source code and explanations also reference the idea of the delta hyper-parameter, which is a very small scalar all points are multiplied by to keep them from "exploding" or going to infinity. Depending on the value of delta, the attractor may look different than expected, through no fault of the program. This is something I need to consider when evaluating the success of the program, as I will compare the images generated by my program to existing images of the attractors to determine the success of the program. I would like the user to be able to edit this delta value in the preferences window.

The code also discards the first couple thousand points of an attractor (a different amount per attractor) as these points form a "tail" outside the basin of the attractor. This tail is not part of the basin of attraction, so should not usually be included in the output image of a solution. This is something I should consider when creating my solution, however I would like the user to be able to choose to include or not include the tail in the preferences.

Overall, I think this rendering method could be highly effective to generate detailed images, however due to the high space complexity of the algorithm it could fail on machines with less than the required ram to store a list of all points, especially when points for attractors are in the billions. This issue, however, is only present because all points are generated then simultaneously rendered. A similar method where a point is generated, rendered and then that point is used to generate the next point and the first point dropped out of memory could potentially remedy this issue, at the cost of time complexity.

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.

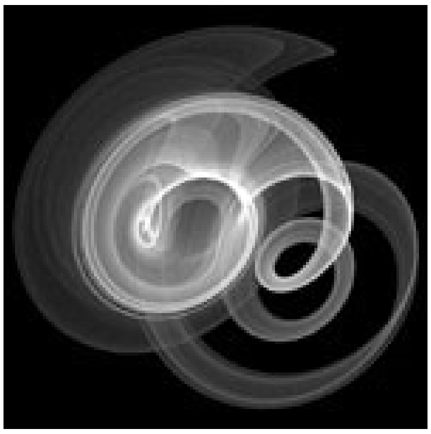


Figure 10: An example output file

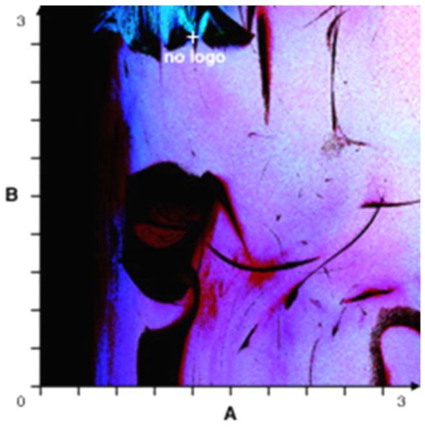
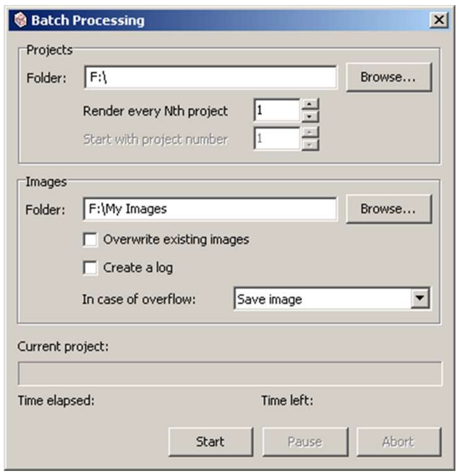


Figure 10: The search window with custom parameter axis.



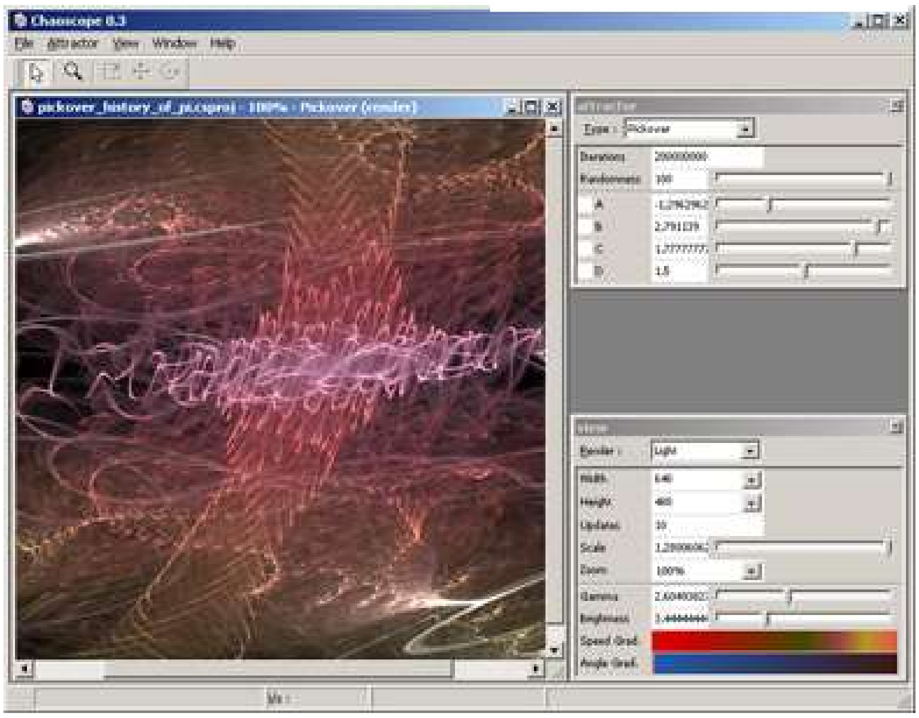
Batch Processing

Projects
Folder: F:\
Render every Nth project: 1
Start with project number: 1

Images
Folder: F:\My Images
☐ Overwrite existing images
☐ Create a log
In case of overflow: Save image

Current project:
Time elapsed:
Time left:

Figure 10: The batch rendering menu



Chaoscope 0.3
pickover_history_of_pi.caproj - 100% - Pickover (render)

Attractor
Type: Pickover
Iterations: 200000000
Randomness: 100
A: -1,2962962
B: 2,791139
C: 1,2777777
D: 1.5

View
Render: Light
Width: 640
Height: 480
Updates: 10
Scale: 1,2000000
Zoom: 100%
Gamma: 2,6040000
Brightness: 1,4444444
Speed Grad.
Angle Grad.

Figure 10: The program in use.

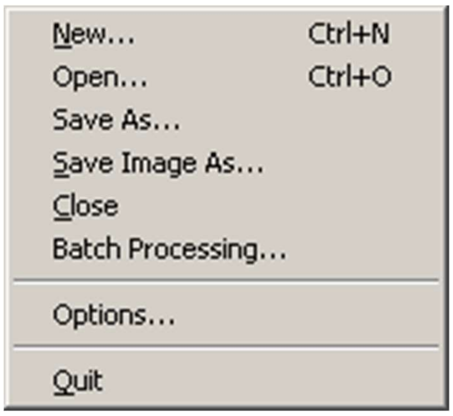


Figure 10: The File menu

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 and graphical 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:

- Inability to use user defined equations
- Colormaps are hard to use and are light based.
- Complicated UI that is difficult to use without mathematical background.

Technological Analysis

The program has 5 rendering modes to choose from. The "Gas" rendering mode (Fig. 6) is based on "pixel accumulation". The pixel grid is projected into space as a lattice, and extended into the Z dimension infinitely. Whenever the orbit is updated, the "brightness" of the lattice point (corresponding to a pixel) is increased by one. Then all pixel values are normalised and a gradient is applied, usually black to white as in Figure 6. Each iteration is one update of the orbit and one pixel brightness increase. The gas rendering mode was designed to be used on old CRT monitors, and as such has parameters that make the images more clear specifically on those CRT monitors, such as "Gamma" and "Contrast". However, these parameters are no longer as necessary as modern monitors do not suffer from low brightness and contrast and therefore do not need to be adjusted for.

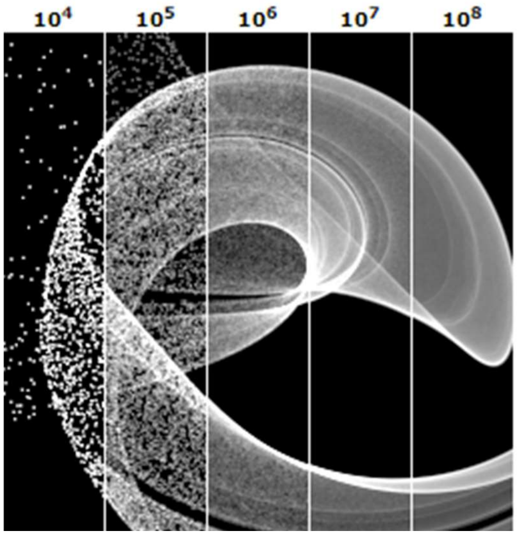


Figure 11: The gas rendering mode at different numbers of iterations

The "Liquid" rendering mode is the same as the Gas rendering mode, except adds depth and opacity via the use of a "Z-Buffer". The Z-Buffer rendering splits the 3D space into a lattice of width and height being the resolution of the screen, and the depth being some value. Then, the usual pixel accumulation algorithm is run, except pixels are discarded if a pixel has a lower Z value (i.e. is in front of that pixel from the cameras perspective.). Opacity means that each pixel is given an opacity value, proportional to the brightness value. When this value is less than one, pixels behind this pixel contribute some proportional amount to the brightness of the pixel in front rather than being discarded.

The "Light" rendering mode is similar to the Gas mode, but instead of assigning brightnesses to pixels, we assign colors to points in space, and then to the relevant pixels. This means the color of a pixel encodes data about the points that pass through that space, usually being the speed and angle. This can be set to various gradients to change the look of the final image.

The "Plasma" rendering mode is similar to the "Light" rendering mode, but makes use of a Z buffer and opacity.

Finally, the "Solid" rendering mode is similar to the Liquid mode, except the accumulation value defines only opacity, and the Z-Buffer is rendered using a ray-tracing method. This allows the render to look like a solid surface, with roughness, diffusion, highlights, reflections and more.

The Solid rendering mode is not useful for my project as there will be no Z-Buffer as the project will only deal with the X and Y dimensions. The same therefore goes to the Liquid, Light and Plasma modes too, as they also utilize a Z-Buffer which would not have any affect on the rendering of images by my program. However, the Gas rendering mode could be a highly effective approach for rendering in my project. The algorithm is simple and slowly increases in accuracy and clarity as it is run, and can be used in 2D easily. The method, due to its iterative nature, would also work on slower machines as it can simply be rendered at a lower number of iterations, as well as being able to effectively render previews by limiting the resources available and iterating slowly. This method also circumvents the space complexity issues presented by the algorithm used in *Visions of Chaos*. As such, I will give consideration to this as a rendering method.

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.

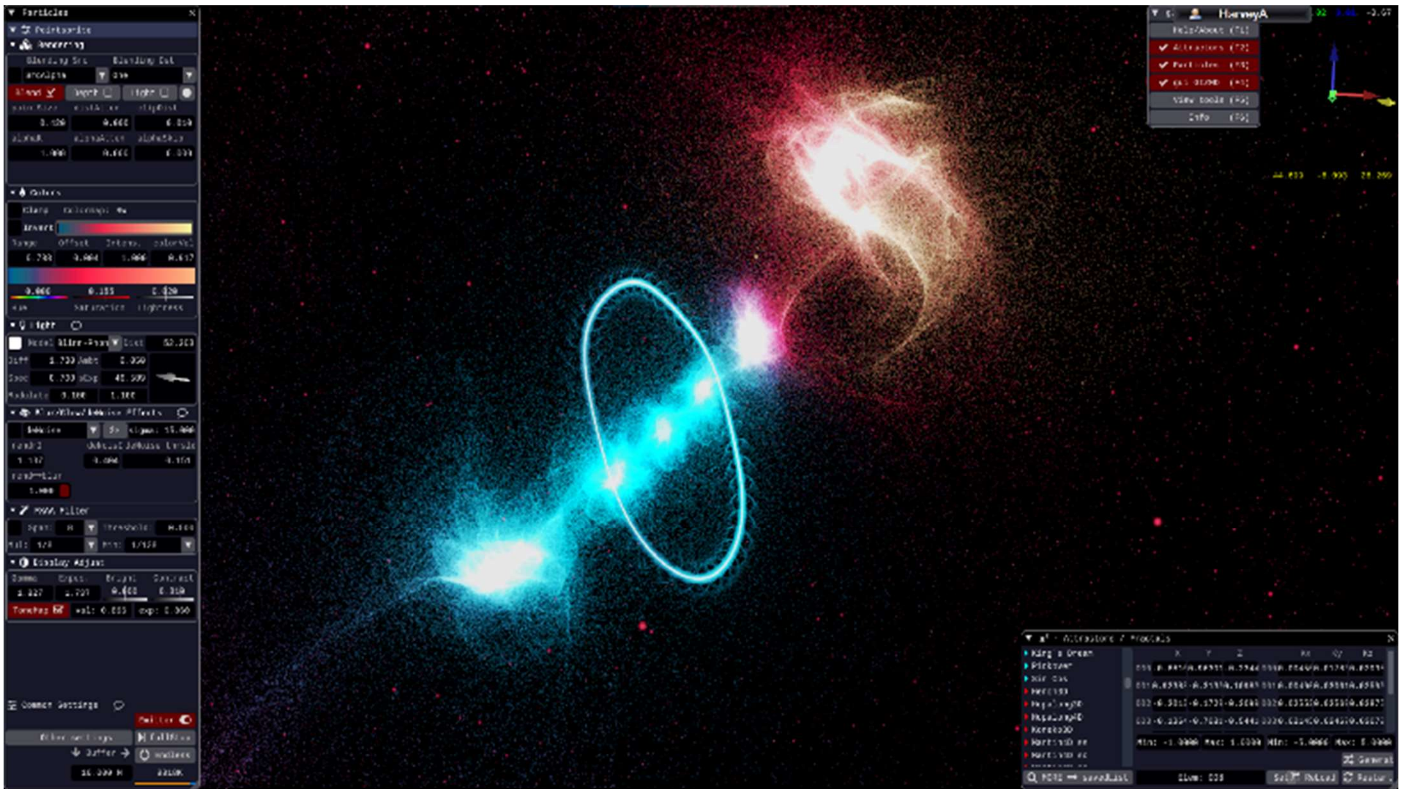


Figure 12: wglChAoS.P rendering a fractal. On the left is shown the rendering settings

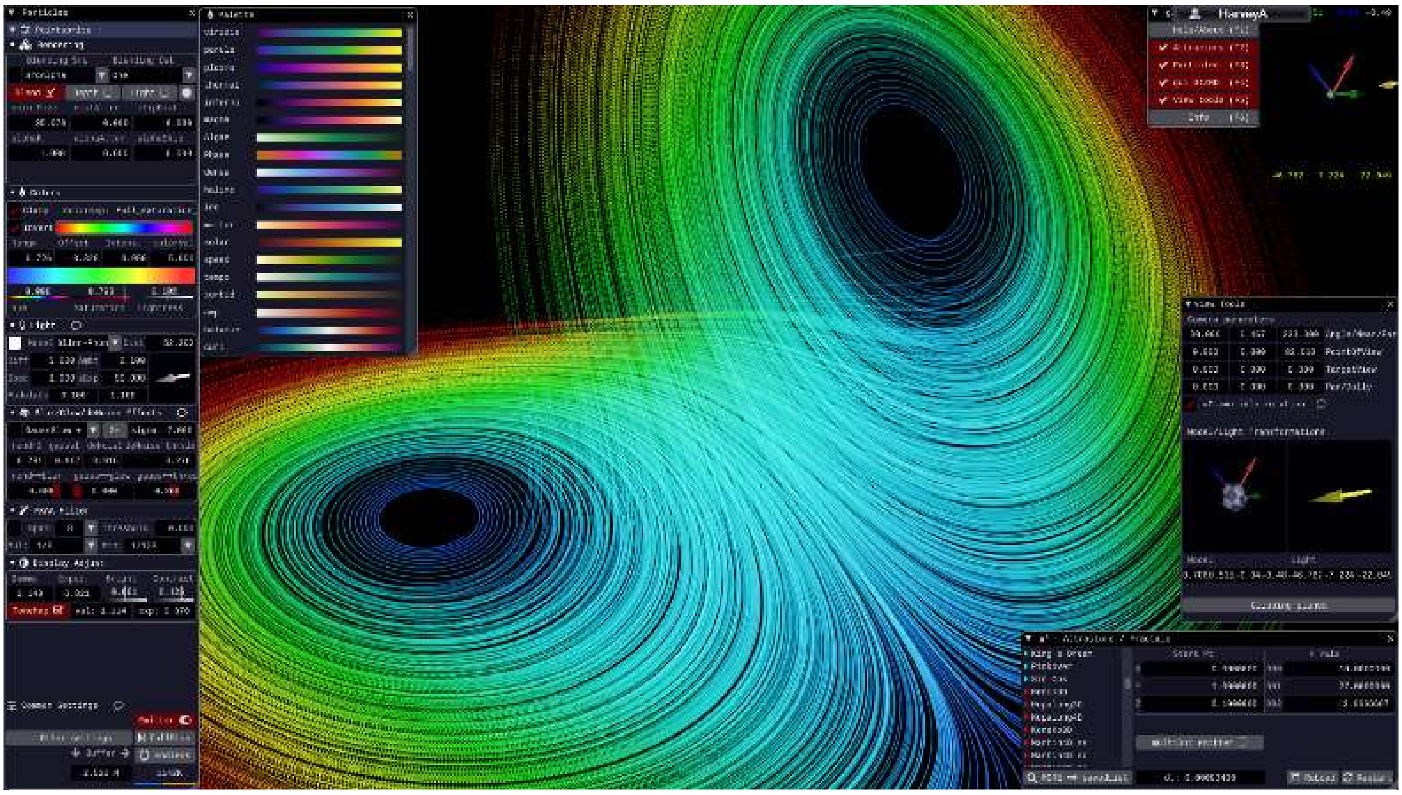


Figure 13: wglChAoS.P rendering the Lorenz attractor. Rendering, colour, and viewport settings are shown.

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 and complex.

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. This is similar to the Gas approach in *Chaoscope*, but does not rasterize a point into its closet pixel, but rather renders the point as a sphere with some radius. Due to the high resolution of modern screens, this may be a better approach as the human eye struggles to differentiate pixels on a modern monitor, so colouring pixels could lead to eye strain as the user cannot properly see individual pixels, but could easily see a particle of fixed radius, spanning multiple pixels.

Furthermore, the approach is suited to both high and low end machines as lower emitting rates and less emitters can speed up rendering (at the cost of lower clarity and slower increase in clarity) on low end machines, and higher emitting rates and more emitters can be used on higher end machines to utilise the more powerful graphics processors and therefore speed up rendering.

As such, this emitter rendering approach could be highly effective for my project, and may be well suited to the stakeholders needs. The emitter approach is similar to the approach in *Chaoscope*, and also circumvents the memory issue in *Visions of Chaos*, while also being able to reduce time complexity using multiple emitters and multithreading. As such, I think this is the most suited approach to the rendering problem.

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 $\dot{x}$ and $\dot{y}$ 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. This limitation is not major as programs already exist to visualise 3d systems, whereas programs to deal with 2d systems do not, so this limitation could be considered a benefit or limitation.

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. However, this limitation is only present in the proof-of-concept that I will be developing, and a fully developed solution might be coded in a lower level language and therefore avoid this limitation.

Mathematical

There will be some mathematical limitations on the solution. For example, the libraries I use for the various mathematical functions might not be as precise as they could be. This could lead to significant variations in output due to the chaotic nature of the maths – small changes in input lead to big changes in output. This limitation could be significant as it may cause unexpected mathematical behaviour and lead to incorrect analysis. This limitation cannot be entirely circumvented either, but I could use more precise libraries and lessen the error margin, however that could require using libraries I am not comfortable using, or potentially coding my own.

Rendering

Modern monitors can only show so much detail. Strange attractors have an infinite amount of detail due to their fractal nature. As such, modern computers will never be able to fully capture all of the detail of a strange attractor, and could potentially lose out on clarity and detail. My project will suffer from this limitation as it also is intended for use on a modern computer. However, this limitation is unavoidable, so is not an issue with my program but rather with the problem itself.

Analysis - Requirements

Stakeholder Requirements

Requirement	Explanation
Lightweight, graphical interface	The GUI needs to be lightweight and easy to understand so that less technical users looking to use it to relax rather than for maths can easily operate the program without needed mathematical or technical knowledge.
Running on both Linux and Windows	One of my stakeholders uses Windows, the other uses Linux, so the program will have to run on both.
Output of common filetypes such as PNG.	The output files will need to be viewed, edited and shared easily so use of a common filetype is required
Saving and loading projects.	Saving and loading projects is very important to Mark as he may want to revisit areas of interest and it would be difficult to write down or remember the large amount of parameters the system may be using.

Functionality Requirements

Requirement	Explanation
Rendering still images of a system.	The program should be able to render still images of a system's phase space given its equations and a viewport.
Entering custom equations for a system.	The user should be able to enter their own custom governing equations
Moving around the viewport to render different areas.	The user should be able to move, rotate and scale the viewport as they see fit.
Editing parameters of a system.	The user should be able to edit the parameters of a system of equations.
Settings menu.	There should be a built in graphical settings menu.
Changing colours of the image.	The user should be able to change the colormap of the image.
Search window.	The user should be able to use a search window to find areas of interest in phase space.
Range of built-in mathematical functions.	The user should be able to utilise a range of different mathematical functions when entering their custom equations.

Hardware and Software Requirements

Requirement	Explanation
A computer with standard periphery	The computer needs to have a mouse and keyboard to use the GUI.
Python 3+ interpreter with libraries	The program will be written in python and will need libraries:
	PyOpenGl – for image processing
	TKinter – for GUI rendering.
	scikit-video – to process videos.
Image displaying software	The user will need software such as nomacs to view the output images.
Video displaying software	The user will need software such as VLC to view the output videos
Windows or Linux	The user will need either a windows operating system or a linux operating system.

Analysis – Success Criteria

SC		Name		SRC	Description	Justification	Evidence
1	a	GUI Windows	Main window	Me	The main window where parameters, colormaps, equations and such will be entered, saved and loaded.	The program needs the main window to function.	Screenshots of the different windows, and videos of their usage.
	b		Settings window		The settings window where output filetype, rendering settings and such will be entered, saved and loaded.	A settings window allows for decluttering and easy setting changing.	
	c		Preview window	SH	The preview window where a lower resolution preview of the render will be shown.	A preview window allows the user to see the effect of their changes before rendering and was requested by a stakeholder.	
	d		Search window		The search window that allows the user to search through either parameter or XY space and find points of interest.	A search window allows the user to more easily find interesting viewports, and was requested by Mark.	
2	a	GUI Usage	Tooltips	SH	Tooltips and lightweight, intuitive GUI controls used so that the user can easily understand and use elements of the GUI.	A new end user will feel lost in a cluttered, undocumented UI. Also, Isabelle requested tooltips.	Video of usage showing a lack of clutter and clear controls with tooltips.
	b		Ease of use	Me			
3	a	Command Line Interface	CLI operation of the program	SH	The program should also be able to be operated entirely from the command line or terminal.	Stakeholder Mark expressed interest in being able to batch render.	Screenshots of command-line operation working.
	b		CLI batch rendering		The user should be able to utilise batch rendering while operating the program from the command line.		
4	a	Saving/Loading	Saving project files	SH	Saving and loading project files that restore parameters, equations, viewport, colormap, etc.	Saving and loading project files was talked about with stakeholders.	Videos of saving and then loading project and config files.
	b		Loading project files	SH			
	c		Saving config files	Me	Saving and loading config files that restores rendering settings, preferences, etc.	Saving and loading config files allows for users to share and transfer their rendering settings and preferences.	
	d		Loading config files				

5	a	Rendering	Rendering Stills	Me	Rendering still images given a viewport, colormap and a system of equations.	This is required for the basic function of the program.	Example inputs and outputs.
	b		Rendering Videos		Rendering videos given a viewport, colormap and system of equations and a variable to change with time.	This would allow for more interesting and soothing visuals.	
	c		Supersampling		Rendering stills at a higher resolution then downscaling for a higher fidelity	This allows for higher fidelity and clarity in images, for analysis.	
	d		Anti-aliasing		Using an anti-aliasing algorithm on the rendered stills to reduce or remove "jaggies"		
	e		High-res mode		An optional "High-res" mode that renders in 4k and up.		
	f		Filetype support		Support for at least PNG and JPG output filetypes.	End users may want to share their outputs, and having support for multiple filetypes makes that easier.	
	g		Colormaps	SH	Colormaps can be applied to a render that colors points differently based on a metric that applies to all points.	Isabelle asked for this feature.	
6	a	Maths	Entering custom equations.	SH	Entering custom equations to govern a system.	Stakeholder Mark asked for this feature for analysis purposes.	Example custom equations, screenshots of them being entered and their outputs
	b		Support for a variety of mathematical functions.	Me	Support for functions such as sine, cosine, natural log, etc.	This would allow Mark to enter any equations that he would want to analyse.	
7		Parameters		Me	Support for attractors with parameters, as well as GUI elements that allow them to be changed.	Some attractors have parameters. Being able to change them allows for those attractors to be used.	Screenshots of changes in parameters via GUI then a change in the output.
8		Built-in example projects		Me	Some project files are built into the software.	New users can look at these example projects to learn the software or because they create soothing visuals	Screenshots of example project files and their default outputs.
9		Support for multiple operating systems.		SH Me	Support for Windows-based and Linux-based operating systems	Stakeholder Mark uses Linux, I use both and Isabelle uses windows.	Screenshots of the program running on both operating systems.

10	Accuracy	Me	The program should accurately create images of the attractors, with the same image being produced given the same equation and parameters. For example, the Lorenz attractor should look like the Lorenz attractor and be mathematically identical to other renders of the Lorenz attractor, every time it is rendered.	Without accuracy, the program would be pointless for use by Mark, and could confuse new users wanting to test “classical” attractors.	Screenshots of attractors, then compared to existing images of those attractors. Note that changes in rendering settings and delta value may lead to slight variations, but this is acceptable as they are mathematically identical.

Design - Decomposition

Design – Structure

Design – Algorithms

Design – Features

Design – Key Things

Design – Testing Data

Design – Further Data