PROJECT PROPOSAL

Integrative Project in Computer Science and Mathematics (420-204-RE)

Brick by Brick (CiHao Zhang, Lucas Chauveau, Qian Qian)

February 3rd, 2025

PROJECT IDEAS

PROJECT IDEAS

Project Ideas	Description
CiHao Zhang *idea #1: Fractal Generator*	A fractal generator that allows various domains to implement fractals to analyze repeating/recurring patterns.
Lucas Chauveau *idea #2: Wi-Fi Access Point Locator*	A graphical application to let pen testers find the approximate physical position of an access point.
Qian Qian *idea #3: Scientific Programmable Calculator*	A calculator program that has an arithmetic calculator and a graphing calculator integrated. This calculator also supports programming, so the user can interactively create a series of calculations to perform over a set of inputs, and the calculator can execute this ”script” automatically according to user inputs.
Chosen Idea: *Scientific Programmable Calculator*	

FRACTAL GENERATOR: CONCEPT & ASPECTS

- Concept: Our fractal generator uses generative algorithms that demonstrates various fractals like the Mandelbrot Set, Julia Set, and other fractal patterns using mathematical equations.
- What problem are we solving? Fractals are essential patterns to detect in many leading fields of the world, such as pattern trading (shorting), health care and signal antennas for connectivity. Our generator allows these domains do visualize the patterns in order to analyze and compare certain patterns with features such as zooming, panning and highlighting tools to focus on specific patterns.

FRACTAL GENERATOR: PARAMETERS, INPUTS & OUTPUTS

- Notably, this application will take into its system a given function, that will be iterated recursively across an x amount of iterations to assure the user's desired effect. This means that we will take in both the equation of the function and the amount of times to repeat the given function. This means that given the function, for example, $Z_{n+1} = Z_n + C$, iterated over 20 times, will generate the given function, composed of each of its individual shapes stacked on each other to create the allure of a fractal. This can then be compared side by side by creating a new instance or to be zoomed/panned.

FRACTAL GENERATOR: FEASIBILITY AND TASK DISTRIBUTION

- In terms of frameworks, we will obviously use Javafx for the main application, we'll use SQLite to save pngs of fractals, and ChartFx to draw the fractal on a canvas, using animations and paint from JavaFx.
- In terms of task distribution, we plan to integrate generative and drawing algorithms, aswell as the database system working independently, as they are our three defining features of this application. We plan to make them functional by deliverable 1, and perfected in deliverable 2, meaning that they are adaptable to many cases and free of all bugs. Deliverable 2 will complete nearly 70% of the application, with a presentable UI (not perfect) and a functioning, complete application. Deliverable 3 completes our project. We must deliver a fulfilling application that is both visually pleasing and contains all features implemented while being free of bugs.

FRACTAL GENERATOR: INDIVIDUAL PARTS

- Deliverable 1:

Cihao : Setting up database & connect to parameters for retrieval & outputs

Lucas: Drawing Algorithm that displays fractals based on points or images

Qian : Generate the fractal per iteration with either points or images

- Deliverable 2:

Cihao : ChartFX canva & Comparison feature

Lucas : UI / UX design for main app & comparison / Zooming & panning

Qian : Design of graphical components & Comparison feature

- Deliverable 3:

All : Bug fixes & UI improvements

WIFI ACCESS POINT LOCATOR: CONCEPT

- For the physical aspects, we need to have an access point available to track.
- For the math behind it, we first gather the signal noise in dB and we have to use the Signal-to-Noise ratio to convert it in signal strength. Once we have the signal strength, we need to triangulate a few positions and draw a graph to show the approximate position. We just need to take into consideration that the signal-to-noise ratio is a logarithmic function.

WIFI ACCESS POINT LOCATOR: CONCEPT ASPECTS

- When penetration testing, it can be very useful to locate an access point to test physical vulnerabilities alongside the digital ones.
- This project would allow the user to find the approximate position of the access point based off a few test points. As far as we can tell, no other project has been made to achieve this so it would be quite a feat.
- For the different variables, we would let the user choose the amount of test point to base the approximation on (min 3, max ??) and the amount of time spent on each test point to get a better strength reading per test point.

WIFI ACCESS POINT LOCATOR: INPUT/OUTPUT

- For the user, the typical input will be the amount of position they want to test to have as good of a reading as possible
- For the program, The input will be the reading at each of these position in the form of signal strength(%)
- The output will be a nice chart, showing circles that correspond to the test positions (where the radius is the uncertainty in signal strength) and the approximate point where the router is located.

WIFI ACCESS POINT LOCATOR: FEASIBILTY AND TASK DISTRIBUTION

- For the user interface, JavaFX will be utilized to implement our GUI and our graph would be generated using matplotlib4j, an open source library ported from the popular matplotlib plotting library in python.
- The project is rather feasible with the timeline since most of the intricate parts would be to develop our own scripts to read the computer's interface.

- Chow: Mathematical formulas to convert noise to relevant numbers and triangulation algorithms
- Qian: Cross-Platform implementation of the project
- Lucas: Making scripts to find nearby access points using the computer's interface and optaining the signal strength.

SCIENTIFIC PROGRAMMABLE CALCULATOR: CONCEPT & ASPECTS

- The main concepts behind the project consist math concepts learned throughout the secondary school math courses and Calculus courses, e.g.: basic calculation, trigonometric functions, function drawing, derivatives and integrals, etc.
- It is often useful to have an all-in-one integrated multifunctional calculator. Even though there are a lot of calculators around that runs on computer, there are rarely one that has all the features like graphing and programming. So, putting all these in the same package can be beneficial.

SCIENTIFIC PROGRAMMABLE CALCULATOR: CONCEPT & ASPECTS

- The main problem this program solves is the need of having to do basic calculations, and the corresponding solution is to provide a arithmetic calculator, with supports for functions like sin, cos, tan, log, etc. It is partially mathematical as everything that need to be properly calculate according to the input, the program also has to correctly interpret the input and turn them into correct calculation steps in the right order.
- Secondly, the graphing calculator part of the program handles the need of visualizing a certain functions. The mathematical aspect of this feature is also calculating each points of the graph. For this feature, the user should be able to adjust various settings, like the displaying range over the axis, to change the user interface showing the graph.
- Lastly, the programming calculator feature is for addressing the need of applying the same series of calculations over different sets of numbers. This feature contains mathematical concepts, but also programming concepts as codes need to be accurately generated and parsed to represents the calculation process.

SCIENTIFIC PROGRAMMABLE CALCULATOR: TYPICAL INPUTS/EXPECTED OUTPUTS

- For this calculator app, the main types of typical inputs are the formulas that needs to calculate, and the numbers needed for the calculation. The expected outputs in this case then is the final result of the calculation.
- For the graphing calculator, the input should be a function, with a parameter x, and then the output is that very function drawn as a graph and visually presented.
- Finally, for the programmable feature, the input should consist of formulas and empty variables (user input) for which the user can plugin their own values for the final calculation. And the output should be a file describing the calculation, and this file should be able to be read by the calculator again to perform the calculation according to the user input.

SCIENTIFIC PROGRAMMABLE CALCULATOR: FEASIBILITY

- Most of the required features are feasible just with Java and JavaFX. On the other hand, libraries like ChartFX (https://github.com/fair-acc/chart-fx) and LatexView (https://github.com/egormkn/latexview) makes the displaying of function graph (for graphing calculator) and the mathematical formula the user inputted.
- On the other hand, for the programming feature, even though a “language interpreter” is needed, books like Crafting Interpreters (https://craftinginterpreters.com/) that provides detailed instructions are very well available online. Considering that the book only has 10 short chapters for building the language interpreter, and that a lot of the features (like control flow, classes, inheritance, etc.) of the language presented in the book is not needed, the process for building this feature should not take very long even though a designed from scratch “language” is still needed for better fitting the mathematical needs. The other parts of the projects should also not take a very long time since they are not technically very difficult and is done multiple times, and only the part where derivatives/integrals are calculated needs more in detail research.

SCIENTIFIC PROGRAMMABLE CALCULATOR: INDIVIDUAL PARTS

- Part 1 (Lucas): The arithmetic calculator: keyboard for typing in the equation or the formula to calculate (UI), building the right expression for LaTeX or our own calculator programming language for display and calculation.
- Part 2 (Chow): The graphing calculator: the UI for entering formulas (functions) to draw and the drawing of the graph on a UI element (and its relevant logics).
- Part 3 (Qian): The programmable aspect of the calculator: the design of a simple “language” that can represent a series of calculators or calculator actions, and a language interpreter to read the generated expressions to perform the coded actions.
- Integration of parts: the ultimate goal of the project would be allowing the user to interactively create/program the calculator with the arithmetic/graphing calculator, that is: setting the variables that the user needs to provide, write the formulas needed to perform necessary calculations, etc. Then, the language interpreter would be used to read and execute the coded script and show the result on one of the calculator, based on the user inputs made available.

CHOSEN IDEA: *SCIENTIFIC PROGRAMMABLE CALCULATOR*

- We chose this idea because we can guarantee we will come to a finished products. The other options either have next to no documentation or have complex mathematics behind that make them hardly a good option for a project such as this one. The calculator will let us express the skills we have learnt so far but also get us tinkering about the code implementation of mathematical functions. Overall this idea makes it easy to split the task amongst all the team members while making progression steady.