fract'ol

Computer Graphics Fractals

Summary: This project involves creating graphically beautiful fractals.

Version: 5.0

Contents

Chapter I

Foreword

Here's what Wikipedia has to say on **hydraulic fracturing**:

"Hydraulic fracturing," is the targeted disruption of geological formations with low permeability by means of injection under high pressure of a fluid to micro-cracking and crack the rock. This fracturing can be performed near the surface or at depth (over 1 km or more than 4 km in the case of shale gas) and from vertical wells, sloped or horizontal.

This relatively old technique (1947), developed for conventional oil deposits, is renewed by its association with horizontal drilling (developed from 1980). It is the gradual mastery of the economic viability of this association for non-conventional deposits, which guided the recent development of the operation of these: it made available formerly inaccessible resources, or which have been exploited at exorbitant costs and slowly.

It is performed by fracturing the rock by a mechanical "stress" using a fluid injected under high pressure from a surface drilling, to increase the macro porosity and less the micro porosity. The fluid could be the water, a slurry or a technical fluid whose viscosity was adjusted.

This project is not called *fract'oil* and accordingly has no relation to hydraulic fracturing.

Chapter II

Introduction

The term *fractal* was first used by mathematician Benoit Mandelbrot in 1974. He based it on the Latin word *fractus* which means "broken" or "fractured".

A fractal is an abstract mathematical object, such as a curve or a surface, whose pattern remains consistent at every scale.

Various natural phenomena, such as the Romanesco cabbage, exhibit fractal features.

Now, it's your turn to generate some magnificent fractals!

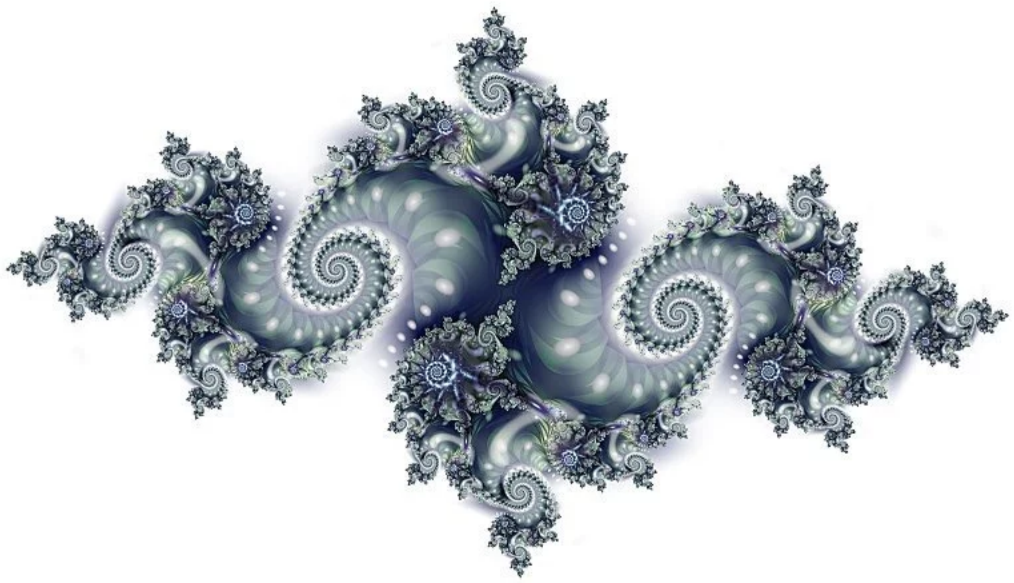
Chapter III

Objectives

It's time for you to create a basic computer graphics project!

You will use the school's graphical library, the `MiniLibX`. This library was developed internally and includes basic necessary tools to open a window, create images and deal with keyboard and mouse events.

This will be an opportunity for you to become familiar with the `MiniLibX` library, discover or use the mathematical concept of **complex numbers**, explore computer graphics **optimization**, and practice **event handling**.



Chapter IV

Common Instructions

- Your project must be written in C.
- Your project must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check, and you will receive a `0` if there is a norm error.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc.) except for undefined behavior. If this occurs, your project will be considered non-functional and will receive a `0` during the evaluation.
- All heap-allocated memory must be properly freed when necessary. Memory leaks will not be tolerated.
- If the subject requires it, you must submit a `Makefile` that compiles your source files to the required output with the flags `-Wall`, `-Wextra`, and `-Werror`, using `cc`. Additionally, your `Makefile` must not perform unnecessary relinking.
- Your `Makefile` must contain at least the rules `$(NAME)`, `all`, `clean`, `fclean` and `re`.
- To submit bonuses for your project, you must include a `bonus` rule in your `Makefile`, which will add all the various headers, libraries, or functions that are not allowed in the main part of the project. Bonuses must be placed in `_bonus.{c/h}` files, unless the subject specifies otherwise. The evaluation of mandatory and bonus parts is conducted separately.
- If your project allows you to use your `libft`, you must copy its sources and its associated `Makefile` into a `libft` folder. Your project's `Makefile` must compile the library by using its `Makefile`, then compile the project.
- We encourage you to create test programs for your project, even though this work **does not need to be submitted and will not be graded**. It will give you an opportunity to easily test your work and your peers' work. You will find these tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to the assigned Git repository. Only the work in the Git repository will be graded. If Deepthought is assigned to grade your work, it will occur

after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter V

AI Instructions

● Context

During your learning journey, AI can assist with many different tasks. Take the time to explore the various capabilities of AI tools and how they can support your work. However, always approach them with caution and critically assess the results. Whether it's code, documentation, ideas, or technical explanations, you can never be completely sure that your question was well-formed or that the generated content is accurate. Your peers are a valuable resource to help you avoid mistakes and blind spots.

● Main message

- ☛ Use AI to reduce repetitive or tedious tasks.
- ☛ Develop prompting skills — both coding and non-coding — that will benefit your future career.
- ☛ Learn how AI systems work to better anticipate and avoid common risks, biases, and ethical issues.
- ☛ Continue building both technical and power skills by working with your peers.
- ☛ Only use AI-generated content that you fully understand and can take responsibility for.

● Learner rules:

- You should take the time to explore AI tools and understand how they work, so you can use them ethically and reduce potential biases.
- You should reflect on your problem before prompting — this helps you write clearer, more detailed, and more relevant prompts using accurate vocabulary.
- You should develop the habit of systematically checking, reviewing, questioning, and testing anything generated by AI.
- You should always seek peer review — don't rely solely on your own validation.

● Phase outcomes:

- Develop both general-purpose and domain-specific prompting skills.
- Boost your productivity with effective use of AI tools.
- Continue strengthening computational thinking, problem-solving, adaptability, and collaboration.

● Comments and examples:

- You'll regularly encounter situations — exams, evaluations, and more — where you must demonstrate real understanding. Be prepared, keep building both your technical and interpersonal skills.
- Explaining your reasoning and debating with peers often reveals gaps in your understanding. Make peer learning a priority.
- AI tools often lack your specific context and tend to provide generic responses. Your peers, who share your environment, can offer more relevant and accurate insights.
- Where AI tends to generate the most likely answer, your peers can provide alternative perspectives and valuable nuance. Rely on them as a quality checkpoint.

✓ Good practice:

I ask AI: "How do I test a sorting function?" It gives me a few ideas. I try them out and review the results with a peer. We refine the approach together.

✗ Bad practice:

I ask AI to write a whole function, copy-paste it into my project. During peer-evaluation, I can't explain what it does or why. I lose credibility — and I fail my project.

✓ Good practice:

I use AI to help design a parser. Then I walk through the logic with a peer. We catch two bugs and rewrite it together — better, cleaner, and fully understood.

✗ Bad practice:

I let Copilot generate my code for a key part of my project. It compiles, but I can't explain how it handles pipes. During the evaluation, I fail to justify and I fail my project.

Chapter VI

Mandatory part

Program Name	`fractol`
Files to Submit	`Makefile, *.h, *.c`
Makefile	`NAME, all, clean, fclean, re`
Arguments	`The type of fractal to display and any other option available`
External Function	• `open, close, read, write, malloc, free, perror, strerror, exit.` • `All functions of the math library (-lm compiler option, man 3 math).` • `All functions of the MiniLibX library.` • `gettimeofday()` • `ft_printf or any equivalent YOU coded.`
Libft authorized	`Yes`
Description	`This project is about creating a small fractal exploration program. First, you need to understand what a fractal is.`

Your project must comply with the following rules:

- You **must** use the `MiniLibX` library. Either the version available on the school machines, or installing it using its sources.
- You have to turn in a `Makefile` which will compile your source files. It must not relink.
- Global variables are forbidden.

VI.1 Rendering

- Your program must offer the **Julia** set and the **Mandelbrot** set.
- The mouse wheel allows zooming in and out almost infinitely (within the computer's limits). This is the very principle of fractals.
- You must be able to create different Julia sets by passing different parameters to the program.
- A parameter is passed on the command line to define what type of fractal will be displayed in a window.
    - You can handle more parameters to use them as rendering options.
    - If no parameter is provided, or if the parameter is invalid, the program displays a list of available parameters and exits properly.
- You must use at least a few **colors** to reveal the depth of each fractal. Experimenting with psychedelic effects is encouraged.

VI.2 Graphic management

- Your program has to display the image in a window.
- Window management must remain smooth (e.g., switching to another window, minimizing, etc.).
- Pressing `ESC` must close the window and quit the program in a clean way.
- Clicking on the cross on the window's frame must close the window and quit the program in a clean way.
- The use of the `images` of the `MiniLibX` library is mandatory.

Chapter VII

Readme Requirements

A `README.md` file must be provided at the root of your Git repository. Its purpose is to allow anyone unfamiliar with the project (peers, staff, recruiters, etc.) to quickly understand what the project is about, how to run it, and where to find more information on the topic.
The `README.md` must include at least:

- The very first line must be italicized and read: *This project has been created as part of the 42 curriculum by <login1>[, <login2>[, <login3>[...]]].*

- A "**Description**" section that clearly presents the project, including its goal and a brief overview.

- An "**Instructions**" section containing any relevant information about compilation, installation, and/or execution.

- A "**Resources**" section listing classic references related to the topic (documentation, articles, tutorials, etc.), as well as a description of how AI was used — specifying for which tasks and which parts of the project.

➠ **Additional sections may be required depending on the project** (e.g., usage examples, feature list, technical choices, etc.).

Any required additions will be explicitly listed below.

`English is recommended; alternatively, you may use the main language of your campus.`

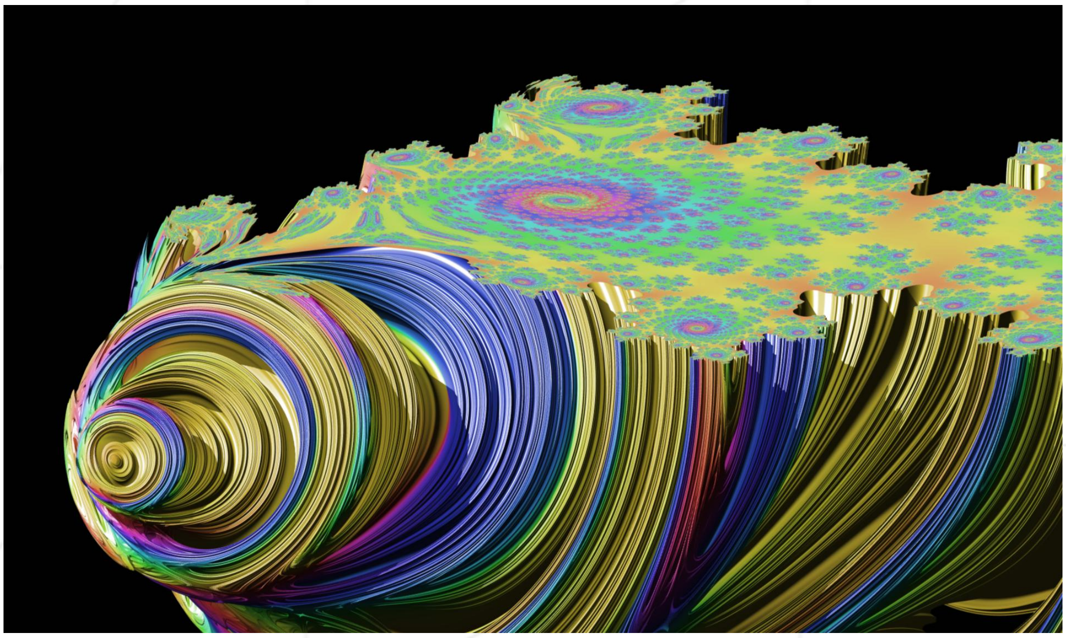
Chapter VIII

Bonus part

Typically, you would be encouraged to develop your own original additional features; however, more interesting graphic projects await you in the future. Don't spend too much time on this assignment!

You will get some extra points with the following features:

- One more different fractal (more than a hundred different types of fractals are referenced online).
- The zoom follows the actual mouse position.
- In addition to zooming, allow moving the view using the arrow keys.
- Make the color range shift.



`The bonus part will only be assessed if the mandatory part is PERFECT. Perfect means the mandatory part has been integrally done and works without malfunctioning. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.`

Chapter IX

Submission and peer-evaluation

Submit your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your files to ensure they are correct.

As these assignments are not verified by a program, feel free to organize your files as you wish, provided you submit the mandatory files and comply with the requirements.

During the evaluation, a brief **modification of the project** may occasionally be requested. This could involve a minor behavior change, a few lines of code to write or rewrite, or an easy-to-add feature.

While this step may **not be applicable to every project**, you must be prepared for it if it is mentioned in the evaluation guidelines.

This step is meant to verify your actual understanding of a specific part of the project. The modification can be performed in any development environment you choose (e.g., your usual setup), and it should be feasible within a few minutes — unless a specific timeframe is defined as part of the evaluation.
You can, for example, be asked to make a small update to a function or script, modify a display, or adjust a data structure to store new information, etc.

The details (scope, target, etc.) will be specified in the **evaluation guidelines** and may vary from one evaluation to another for the same project.