

## SCALE FOR PROJECT FRACT-OL

You should evaluate 1 student in this team

### Introduction





Please comply with the following rules:

- Remain polite, courteous, respectful and constructive throughout the evaluation process. The well-being of the community depends on it.
- Identify with the student or group whose work is evaluated the possible dysfunctions in their project. Take the time to discuss and debate the problems that may have been identified.
- You must consider that there might be some differences in how your peers might have understood the project's instructions and the scope of its functionalities. Always keep an open mind and grade them as honestly as possible. The pedagogy is useful only and only if the peer-evaluation is done seriously.

### Guidelines

- Only grade the work that was turned in the Git repository of the evaluated student or group.
- Double-check that the Git repository belongs to the student(s). Ensure that the project is the one expected. Also, check that 'git clone' is used in an empty folder.
- Check carefully that no malicious aliases was used to fool you and make you evaluate something that is not the content of the official repository.
- To avoid any surprises and if applicable, review together any scripts used to facilitate the grading (scripts for testing or automation).
- If you have not completed the assignment you are going to evaluate, you have to read the entire subject prior to starting the evaluation process.
- Use the available flags to report an empty repository, a non-functioning program, a Norm error, cheating, and so forth.  
In these cases, the evaluation process ends and the final grade is 0, or -42 in case of cheating. However, except for cheating, student are strongly encouraged to review together the work that was turned in, in order to identify any mistakes that shouldn't be repeated in the future.
- You must also verify the absence of memory leaks. Any memory allocated on the heap must be properly freed before the end of execution.  
You are allowed to use any of the different tools available on the computer, such as leaks, valgrind, or e\_fence. In case of memory leaks, tick the appropriate flag.

### Attachments

-  [subject.pdf](#)
-  [minilibx\\_mms\\_beta.tgz](#)
-  [minilibx-linux.tgz](#)
-  [minilibx\\_opengl.tgz](#)

### Preliminary tests

Search for Julia and Mandelbrot sets on Wikipedia, or watch the e-learning videos, to have an idea of what the rendering must look like.

#### Minimal requirements

Does the assignment meet the minimal requirements?

- The repository isn't empty.
- Norminette shows no errors.
- No cheating.
- No forbidden function/library.
- There is no global variable.
- The code compiles with the required options.
- The executable is named as expected.
- During execution, there is no brutal or unmanaged crash (segfault, bus error, and so forth).
- No memory leaks.

✔ Yes

✗ No

### Mandatory part

Evaluate the following points.

#### Graphic management

Is the graphic management functional?

- When the program runs, there is at least a graphic window open.
- Pressing the key ESC exits the program properly (no leaks).
- There is a visual change when using the mouse wheel (even if wrong, this part is about checking the event management only).

✔ Yes

✗ No

#### Julia

How does the Julia set behave?

- Does it looks like it should? Search for the Julia fractal set on the internet if you're not sure.
- Is it possible to zoom in and out and the basic pattern repeats?
- Search for some different Julia sets. You should be able to recreate them by passing different parameters to the program.
- Are there colors to represent the depth of the fractal?

✔ Yes

✗ No

### Mandelbrot

How does the Mandelbrot set behave?

- Does it look like it should?
- Is it possible to zoom in and out and the basic pattern repeats?
- Compare the result with some reference pictures: they all look alike, you can't miss them.
- Are there colors to represent the depth of the fractal?

✔ Yes

✗ No

### Parameters management

Is the parameter's management implemented according to the subject?

Are wrong parameters correctly handled?

(Some optional parameters could be handled, for example for the bonus part.)

✔ Yes

✗ No

### MiniLibX images

Take a look at the code and check whether the student uses the images from the MLX to draw the image instead of putting pixels one by one. :)

✔ Yes

✗ No

## Bonus part

*A lot of nice extras.*

### Zoom follows the mouse

The zoom works where the mouse is and not only at the center of the image.

✔ Yes

✗ No

### Arrows

It must be possible to move the view using the arrow keys. It should also work with the zoom: if an arrow is pressed, the view should move if the user didn't zoom in or out, but also if they did.

✔ Yes

✗ No

### Colors

The color palette is awesome.

- Either you say Ouhaaa because it's very beautiful.
- Either you say Ouhaaa because it's insanely psychedelic.
- Either you say Ouhaaa because the colors change.

✔ Yes

✗ No

### A third fractal

There is at least one more fractal!! Working, nice, and that is actually a fractal different than the two mandatory ones.

✔ Yes

✗ No

## Ratings

Don't forget to check the flag corresponding to the defense

✔ Ok

★ Outstanding project

Empty work

📄 Incomplete work

⚠ Invalid compilation

❏ Norme

📄 Cheat

💥 Crash

⚠ Concerning situation

❏ Leaks

! Forbidden function

💬 Can't support / explain code

## Conclusion

Give this repository a star. ★