SCALE FOR PROJECT FRACT'OL (/PROJECTS/FRACT-OL)

Introduction

Please respect the following rules:

- Remain polite, courteous, respectful and constructive throughout the correction process. The well-being of the community depends on it.

- Identify with the person (or the group) graded the eventual dysfunctions of the work. Take the time to discuss and debate the problems you have identified.

- You must consider that there might be some difference in how your peers might have understood the project's instructions and the scope of its functionalities. Always keep an open mind and grade him/her as honestly as possible. The pedagogy is valid only if the peer-evaluation is conducted seriously.

Guidelines

- Only grade the work that is in the student or group's GiT repository.

- Double-check that the GiT repository belongs to the student or the group. Ensure that the work is for the relevant project and 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 other than the content of the official repository.

- To avoid any surprises, carefully check that both the correcting and the corrected students have reviewed the possible scripts used to facilitate the grading.

- If the correcting student has not completed that particular

project yet, it is mandatory for this student to read the
entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository,
non-functioning program, a norm error, cheating etc. In these cases,
the grading is over and the final grade is 0 (or -42 in case of
cheating). However, with the exception of cheating, you are
encouraged to continue to discuss your work (even if you have not
finished it) in order to identify any issues that may have caused
this failure and avoid repeating the same mistake in the future.

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/3876/fract_ol.en.pdf)

Preliminaries

Here is the fract'ol scale. It includes the required checkups to perform with your own test set. IIf you don't know about this project and want to prep, search at least for Mandelbrot's and Julia's sets on Wikipedia and watch the e-learning videos to have an idea of what it must look like. Git clone and evaluate what's in the repository.

Vital minimum

Is the vital minimum implemented?
- Repository isn't empty.
- Author file is present if required in the subject.
- Norminette shows no errors.
- No cheating.
- No forbidden function/library.
- The projects compiles, the executable is properly named. During
execution there is no brutal or unmanaged crash(segfault, bus error,
etc...) Reminded: With the X11 version of minilibX, an unavoidable X11
error happens if the window is closed using the mouse. No penalty in
that case.

☑ Yes ☒ No

Does it work?

Evaluation of the following points

Graphic management

Is the graphic management accurate?

- When the program runs, there is at least a graphic window.
- ESC key exits properly the program.
- Expose is correctly implemented.
- There is a visual evolution when using the mouse wheel (even if wrong, we're just checking the event management here).
- Something happens visually in at least one of the implemented fractals when moving the mouse without clicking on it.

☑ Yes ☒ No

Julia

How does the Julia set behave?

- Does it looks like the video?
- Does the set evolve when moving the mouse around?
- Is it possible zoom in and out and make the basic pattern repeat?
- Compare with some picture that Google can give you if you search "julia set": most of them should be visible when you move the mouse (colors excluded obviously).
- Are there colors on the area's outskirts?

☑ Yes ☒ No

Mandelbrot

How does the Mandelbrot set behave?

- Does it looks like the video?
- Is it possible zoom in and out and make the basic pattern repeat?
- Compare with some picture that Google can give you if you search "mandelbrot set": they all look alike you can't miss it.
- Are there colors on the area's outskirts?

☑ Yes ☒ No

Additional fractals

Is there a third fractal implemented?

- Is it possible zoom in and out and make the basic pattern repeat?

(It's a fractal basic principle).
- If it's a famous fractal, compare it with some pictures you found on Google.

☑ Yes ☒ No

Parameters management

Is the parameter's management implemented according to the subject? Are wrong parameters correctly handled? (some extensions can be accepted in case of bonuses).

☑ Yes ☒ No

Bonus

A lot of cool stuff

Zoom follows the mouse

Like in the videos, the zoom works where the mouse is and not only at the center of the image.

☑ Yes ☒ No

Arrows

It's possible to move the view window using arrows. It also works with the zoom (if we move on the left for example the image moves slightly to the left, whatever the zoom is).

☑ 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 color changes.

☑ Yes ☒ No

A lot of fractales

There is a whole bunch of other fractals!! (They work, they're cool
and it's for real).

Rate it from 0 (failed) through 5 (excellent)

A lot of additional stuff

There are a lot of additional cool features implemented!

- It's possible to render 3D fractals.
- Several windows and different fractals at the same time.
- Parallel or GPU computing.
- Other features not listed here.

Rate it from 0 (failed) through 5 (excellent)

Ratings

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

✔ Ok ★ Outstanding project

Empty work Incomplete work No author file Invalid compilation

Norme Cheat Crash ⊘ Forbidden function

Conclusion

Leave a comment on this evaluation

Preview!!!