(https://profile.intra.42.fr/searches)

(https://profile.intra.42.fr)

SCALE FOR PROJECT SO_LONG (/PROJECTS/SO_LONG)

You should evaluate 1 student in this team

★

Git repository

git@vogsphere.42paris.fr:vogsphere/intra-uuid-74a9765a-5537-4195-ab5

Introduction

Please adhere to the following rules:

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

- Identify possible dysfunctions in the project of the student or group whose work is being evaluated. Take the time to discuss and debate the problems that may have been identified.

- Consider that there might be 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. Pedagogy is useful only if peer evaluation is done seriously.

Guidelines

- Only grade the work submitted in the Git repository of the evaluated student or group.

- Double-check that the Git repository belongs to the student or students. Ensure that the project is the expected one. Also, check that 'git clone' is used in an empty directory.

- Carefully check that no malicious aliases were used to deceive you and make you evaluate something that is not the content of the official repository.

- To avoid any surprises, and if applicable, review any scripts used to facilitate grading (such as testing or automation scripts) together.

- If you have not completed the assignment you are going to evaluate, read the entire subject before starting the evaluation process.

- Use the available flags to report an empty repository, a non-functioning program, a Norm error, cheating, etc.
In these cases, the evaluation process ends, and the final grade is 0, or -42 in the case of cheating. However, except in cases of cheating, students are strongly encouraged to review the submitted work together to identify mistakes that should not be repeated in the future.

- Verify that there are no memory leaks. Any memory allocated on the heap must be properly freed before the program's execution ends. You are allowed to use any of the tools available on the computer, such as leaks, valgrind, or e_fence. In case of memory leaks, tick the appropriate flag.

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/168864/en.subject.pdf)

minilibx_macos_metal.tgz (https://cdn.intra.42.fr/document/document/35894/minilibx_macos_metal.tgz)

minilibx-linux.tgz (https://cdn.intra.42.fr/document/document/35895/minilibx-linux.tgz)

minilibx_macos_opengl.tgz (https://cdn.intra.42.fr/document/document/35896/minilibx_macos_opengl.tgz)

Mandatory part

Executable name

Execute the `make` command. The project should compile as expected (the Makefile should not re-link). Verify that the executable name is `so_long`. Otherwise, use the "invalid compilation" flag at the end of the scale.

Yes No

Parsing

- Use different maps.
- Test different sizes.
- Test different line sizes.

Also, check that the program returns an error and exits properly when the configuration file is misconfigured (e.g., an unknown character, duplicates, an invalid file path, etc.).

Otherwise, the defense is over. Use the appropriate flag: incomplete work, crash, etc.

Yes No

Technical elements of the display

Evaluate the technical elements of the display.
Check that the level accurately represents the map used as parameter.

- A window must open at the launch of the program.
  It must remain open thoughout the execution.
- Hide all or parts of the window either by using another window or by using the screen's borders. Then, minimize the window and maximize it back. In all cases, the window's content must remain consistent.

Yes No

Basic user events

In this section, evaluate the program's user events. Execute the following three tests. If at least one of them fails, no points should be awarded for this section, and you must move to the next one.

- Click the cross at the top of the window. The window must close, and the program must exit cleanly.
- Press the ESC key. The window must close, and the program must exit cleanly. In this test, you can accept that another key exits the program, for example, Q.
- Press the four arrow keys (it is acceptable to use the WASD or ZQSD keys instead) in any order. Each key press must render a visible result on the window (player's movement).

☑ Yes ☒ No

Movements

In this section, evaluate the implementation of the player's movement. Execute the five following tests. If at least one of them fails, no points should be awarded for this section, and must move to the next one.

- The player's spawning position must be in accordance with the map file.
- Press the arrow keys (it is acceptable to use the WASD or ZQSD keys instead) to move in every direction on the map.
- Is the game playable?

☑ Yes ☒ No

Walls & Sprites

In this section, evaluate the map representation. Execute the following tests. If at least one of them fails, no points should be awarded for this section, and you must move to the next one.

- The wall's texture is correctly placed, and the player cannot go through it.
- The collectible's texture is correctly placed, and the player can pick it by walking on it.
- The Exit texture is correctly placed, and the player can finish the game by walking on it, but only after picking every collectible.
- The player texture is correctly placed and can move in every direction except into the walls.

☑ Yes ☒ No

Counter

In this section, evaluate the gameplay elements. Execute the following tests. If at least one of them fails, no points should be awarded for this section and you must move to the next one.

- A small counter is displayed on the shell that counts how many movements the player makes.
- The counter can be displayed directly on the game screen (see Bonus part).

Yes No

MiniLibX images

Review the code and check whether the student uses MLX images to draw the image instead of plotting pixels one by one.

Yes No

Bonus

Evaluate the bonus part if, and only if, the mandatory part has been entirely and perfectly completed, and the error management handles unexpected or bad usage. In case all the mandatory points were not passed during the

defense, bonus points must be completely ignored.

Enemies

Enemy patrols cause the player to lose if they touch them.

☑ Yes ☒ No

Sprite animation

There is sprite animation. The evaluated student must explain what it is and how they implemented it. Rate according to the quality of the animation.

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

5

Counter on screen

The movement counter is rendered on the screen using texture or mlx_string_put(). Rate according to the visual quality and the quality of implementation.

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

5

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

Leave a comment on this evaluation (2048 chars max)

Finish evaluation

API General Terms of Use (https://profile.intra.42.fr/legal/terms/33) | Declaration on the use of cookies (https://profile.intra.42.fr/legal/terms/2) | Privacy policy (https://profile.intra.42.fr/legal/terms/35) | General term of use of the site (https://profile.intra.42.fr/legal/terms/6) | Internal Rules (https://profile.intra.42.fr/legal/terms/7) | Terms of use for video surveillance (https://profile.intra.42.fr/ | Legal notices (https://profile.intra.42.fr/legal/terms/3)

legal/terms/1)