SCALE FOR PROJECT SO_LONG

You should evaluate 1 student in this team

Git repository

`git@vogsphere.42berlin`

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-linux.tgz minilibx_opengl.tgz

minilibx_mms_20200219_beta.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, and so forth).

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

Yes | No

Technical elements of the display

Time to evaluate the technical elements of the display.
Check that the level is an accurate representation of the map used as parameter.

- A window must open at the launch of the program.
  It must remain open during the whole 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 content of the window must remain consistent.

Yes | No

Basic user events

In this section, let's evaluate the program's user events. Execute the 3 following tests. If at least one of them fails, this means that no points must be awarded for this section and you have to 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 the case of 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 the order of your choice. Each key press must render a visible result on the window (player's movement).

Yes | No

Movements

In this section, let's evaluate the implementation of the player's movement. Execute the 5 following tests. If at least one of them fails, this means that no points must be awarded for this section and you have to move to the next one.

- The player's spawning position must be in accordance with the map file.
- Press the arrows 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, let's evaluate the map representation. Execute the following tests. If at least one of them fails, this means that no points must be awarded for this section and you have to 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, let's evaluate the gameplay elements. Execute the following tests. If at least one of them fails, this means that no points must be awarded for this section and you have to move to the next one.

- There's a small counter displayed on the shell that counts how many movements the player does.
- The counter can be displayed directly on the game screen (see 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

Evaluate the bonus part if, and only if, the mandatory part has been entirely and perfectly done, 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 totally ignored.

Enemies

The enemy patrols cause the player to lose if they touch them.

Yes | No

Sprite animation

There's some sprite animation. The evaluated student has to explain what it is and how they did it.

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

Counter on screen

The counter of movements is rendered on the screen using texture or mlx_string_put().

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 | Invalid compilation | Norme | Cheat

Crash | Concerning situation | Leaks | Forbidden function

Can't support / explain code

Conclusion

Leave a comment on this evaluation

Thanks for explaining your code. Everything worked as expected. Good job !

Finish evaluation