Introduction

Please respect 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 person (or the group) evaluated 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 and only if 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 evaluating and the evaluated students have reviewed the possible scripts used to facilitate the grading.

- If the evaluating student has not completed that particular project yet, it is mandatory for this student to read the entire subject before starting the defense.

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

- Remember that for the duration of the defense, no segfault, no other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag.
You should never have to edit any file except the configuration file if it exists.
If you want to edit a file, take the time to explicit the reasons with the evaluated student and make sure both of you are okay with this.

- 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_opengl.tgz minilibx_mms_20200219_beta.tgz

Mandatory part

Executable name

Check that the project compiles well (without re-link) when you execute the `make` command and that the executable name is `cub3D`.
If not, use the invalid compilation flag at the end of the scale.

Yes | No

Configuration file

Check that you can configure ALL the following elements in the configuration file.
The formating has to be as described in the subject.

- north texture path - NO
- east texture path - EA
- south texture path - SO
- west texture path - WE
- floor color - F
- ceiling color - C
- the map (see subject for the map configuration details)

Also, check that the program returns an error and exits properly when the configuration file is misconfigured (for example an unknown key, double keys, an invalid path...) or if the filename doesn't end with the `.cub` extension.

If not, the defense is over and use the appropriate flag incomplete work, crash...

Yes | No

Technical elements of the display

We're going to evaluate the technical elements of the display. Run the program and execute the following tests. If at least one fails, no points will be awarded for this section. Move to the next one.

- A window must open at the launch of the program.
It must stay open during the whole execution.
- An image representing the inside of a maze must be displayed inside the window.
- Hide all or part of the window either by using another window or by using the screen's borders, then minimize the windows and maximize it back. In all cases, the content of the window must remain consistent.

Yes | No

User basic events

In this section, we're going to evaluate the program's user generated events. Execute the 3 following tests. If at least one fails, this means that no points will be awarded for this section.
Move to the next one.

- Click the red cross at the top left 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, we will accept that another key exits the program, for example, Q.
- Press the four movement keys (we'll accept WASD or ZQSD keys) in the order of your liking. Each key press must render a visible result on the window, such as a player's movement/rotation.

Yes | No

Movements

In this section, we'll evaluate the implementation of the player's movement/orientation inside the maze.
Execute the 5 following tests. If at least one fails, this means that no points will be awarded for this section.

- The player's spawning orientation on the first image must be in accordance with the configuration file, test for each cardinal orientation (N, S, E, W).
- Press the left arrow then the right arrow.
The player's view must rotate to the left then to the right as if the player's head was moving.
- Press W (or Z) then S.
The player's view must go forward and then backward in a straight line.
- Press A (or Q) then D.
The player's view must go to the left and then to the right in a straight line.
- During those four movements, was the display smooth? By smooth we mean is the game "playable" or is it slow.

Yes | No

Walls

In this section, we'll evaluate the walls in the maze.
Execute the 4 following tests. If at least one fails, this means that no points will be awarded for this section.

- The wall's texture vary depending on which compass point the wall is facing (north, south, east, west).
Check that the textures on the walls and perspective are visible and correct.
- Check that if you modify the path of a wall texture in the configuration file, it modifies the rendered texture when the program is re-executed.
- Also check that if you set a non-existent path it raises an error.
- Check that the floor and ceiling colors are well handled when you modify them in the configuration file.

Yes | No

Error management

In this section, we'll evaluate the program's error management and reliability. Execute the 4 following tests. If at least one fails, this means that no points will be awarded for this section.
Move to the next one.

- Run the program using numerous arguments and random values.
Even if the program doesn't require any arguments, it is critical that those arguments don't allernate or create unhandled errors.
- Check that there are no memory leaks. You can use the `top` or `leaks` command in another shell to monitor that the memory use is stable. The memory used must not increase each time an action is made.
- Roll either your arm or your face on the keyboard. The program must not show any strange behaviors and it must stay functional.
- Modify the map. The program must not show any strange behaviors and it must stay functional if the map is well configured, if not it must raise an error.

Yes | No

Bonus

We will look at your bonuses if and only if your mandatory part is excellent. This means that you must complete the mandatory part, beginning to end, and your error management must be flawless, even in cases of twisted or bad usage. So if the mandatory part didn't score all the points during this defense bonuses will be totally ignored.

When I'll be older I'll be John Carmack

Look at the subject bonus part and add one point for each bonus implemented and fully functional.

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