SCALE FOR PROJECT MINIRT (/PROJECTS/MINIRT)

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 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.

- Remember that for the duration of the defence, 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 (https://cdn.intra.42.fr/pdf/pdf/34213/en.subject.pdf)

minilibx_opengl.tgz (/uploads/document/document/5828/minilibx_opengl.tgz)

minilibx_mms_20200219_beta.tgz
(/uploads/document/document/5829/minilibx_mms_20200219_beta.tgz)

Mandatory part

Executable name

Check that the project compiles well (without re-link) when you excute the `make` command and that the executable name is `miniRT`.

✓ Yes ✕ No

Configuration file

Check that you can configure camera, light, the ambient light ratio and simple objects in the configuration file in accordance with the format described in the subject.
Also check that the program returns an error and exits properly when the configuration file is misconfigured or if the filename doesn't end with the `.rt` extension.

If not, the defence is over and the final grade will be 0.

✓ Yes ✕ No

Technical elements of the display

In this section we'll evaluate 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.

- With only one parameter a window must open when launching the program and stay open during the program's whole execution.

- Hide either part of the window or the whole window with another window or the screen's borders, minimize the minirt window to the dock/taskbar and maximize it back.
In every case, the window's content must remain consistant (minirt should not quit and should still display properly its content).

- When you change the window resolution, the window's content must remain consistant.

- Pressing `ESC` or clicking the red cross of the window exits the program properly.

✓ Yes ✕ No

The Basic Shapes

In this section we'll evaluate the 3 basic shapes. Run the program and execute the following 3 tests. If at least one fails, no points will be awarded for this section. Move to the next one.

- Place a sphere at the coordinates {0, 0, 0}. With the camera facing the sphere, display the rendered image. The sphere should be visible and displayed without glitching.

- Place a plane with a 'z' value of null. With the camera facing the plane, display the rendered image. The plane should be visible and displayed without glitching.

- Place a cylinder extending along the y axis. With the camera facing the cylinder, display the rendered image. The cylinder should be visible and displayed without glitching.

✓ Yes ✕ No

Translations and rotations

In this section we'll evaluate that rotation and translation transformations can be applied on the scene's objects. 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.

- Place two spheres at the coordinates {0, 0, 0}, the camera facing those spheres. Then put a translation on one of the two spheres oriented in a direction parallel to the camera's, of a greater distance than the sphere's diameter and display the rendered image. Both spheres should be visible and displayed without glitching.

- Place a cylinder extending along the y axis, the camera facing the cylinder. Then put a 90° rotation (PI/2 radian) along the z axis and display the rendered image. The cylinder should be visible and displayed without glitching.

✓ Yes ✕ No

Multi-objects

In this section we'll evaluate that it's possible to put several object in one scene. 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.

- Place several intersecting objects on the scene, such as for example a sphere and a cylinder, and display the rendered image. Both objects should be visible and displayed without glitching. (especially where both object intersect)

- Execute the same test, but ensure it's possible to place the same object several times, for example two cylinders, two spheres and a plane.

✓ Yes ✕ No

Camera's position and direction

In this section we'll evaluate that the camera conditions of the subject are respected. 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.

- Generate a random scene and place the camera extending along the x axis pointed towards the coordinates {0, 0, 0} and display the rendered image. The scene must be visible and displayed without glitching.

- Generate a random scene and place the camera extending along the y axis pointed towards the coordinates {0, 0, 0} and display the rendered image. The scene must be visible and displayed without glitching.

- Generate a random scene and place the camera extending along the z axis pointed towards the coordinates {0, 0, 0} and display the rendered image. The scene must be visible and displayed without glitching.

- Generate a random scene and place the camera at a random location which isn't on any axis or a diagonal, pointed towards the coordinates {0, 0, 0} and display the rendered image. The scene must be visible and displayed without glitching.

✓ Yes ✕ No

Brightness 1/2

In this section we'll evaluate brightness on the scene's objects. 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.

- Place a sphere at the coordinates {0, 0, 0}, the camera facing the sphere, and put a spot left or right of the camera but positioned in such a way that the sphere will be lit sideways.
Display the rendered image. The sphere should be visible, illuminated and displayed without glitching.

- Place a sphere at some coordinates resulting from a translation, the camera facing the sphere, and place a spot left or right of the camera but positioned in such a way that the sphere will be lit sideways. Display the rendered image. The sphere should be visible, properly illuminated and displayed without glitch. Properly means that the halo of light should be computed after translation not before.

✓ Yes ✕ No

Brightness 2/2

In this section we'll evaluate shadow management generated by the scene's objects. 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.

- Place a vertical spot, a sphere and a plane. The spot lighting the sphere's position to create a sphere shadow on the plane. Put the camera aside so we can see the sphere, the plane and the sphere's shadow on the plane. The shadow must be properly displayed without glitching.

- Put a complex scene together with several objects like on illustration V.6 page 10 of the subject. Shadows must be properly displayed without glitching.

✓ Yes ✕ No

Bonus

We will look at your bonuses if and only if your mandatory part is excellent. This means that your 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 point during this defence bonuses will be totally ignored.

Many bonuses?

One point per bonus.
-Specular reflection.
-Color disruption: checkerboard.
-Colored and multi-spot lights.
-One other 2nd degree object: Cone, Hyperboloid, Paraboloid..
-Handle bump map textures.

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

○