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

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

SCALE FOR PROJECT MINIRT (/PROJECTS

You should evaluate 2 students in this team

★

Git repository

`git@vogsphere.42kocaeli.com.tr:vogsphere/intra-uuid-c6930f2`

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 the 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.

- 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. 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.

- Check carefully 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 together any scripts used to facilitate grading, such as testing or automation scripts.

- If you have not completed the assignment you are going to evaluate, you must 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 any mistakes that should not be repeated in the future.

- You must also 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/151604/en.subject.pdf)

minilibx-linux.tgz (https://cdn.intra.42.fr/document/document/31108/minilibx-linux.tgz)

minilibx_opengl.tgz (https://cdn.intra.42.fr/document/document/31109/minilibx_opengl.tgz

minilibx_mms_20200219_beta.tgz
(https://cdn.intra.42.fr/document/document/31110/minilibx_mms_20200219_beta.tgz)

Mandatory part

Executable name

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

✓ Yes ✗ No

Configuration file

Check that you can configure the 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 defense is over and the final grade will be 0.

✓ Yes ✗ No

Technical elements of the display

In this section, we will 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.

- 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 consistent (minirt should not quit and should still display properly its content).
- Pressing `ESC` or clicking the red cross of the window exits the program properly.

✓ Yes ✗ No

The Basic Shapes

In this section, we will evaluate the three basic shapes. Run the program
and execute the following three tests. If any test fails, no points
will be awarded for this section; proceed with 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 zero. 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 will evaluate that rotation and translation
transformations can be applied to 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}, with the camera facing those spheres. Then apply a translation on one of the two spheres oriented in a direction parallel to the camera's, at 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 apply 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 will evaluate that it's possible to put several objects 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 in 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 objects 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 will 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 that is not 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 will 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 to the left or right of the camera but positioned in such a way that the sphere will be lit from the side. 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 glitching. Properly means that the halo of light should be computed after translation, not before.

✓ Yes ✕ No

Brightness 2/2

In this section, we will 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 should illuminate the sphere 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.
- Assemble a complex scene with several objects, as shown in illustration V.6 page 10 of the subject. Shadows must be properly displayed without glitching.

✓ Yes ✕ No

Bonus

We will consider your bonuses if and only if your mandatory part is excellent. This means that the mandatory part from beginning to end, and your error management must be flawless, eve or bad usage. Therefore, if the mandatory part did not score all the point during this defense, l ignored.

Many bonuses?

One point per bonus.

- Specular reflection.
- Color disruption: checkerboard.
- Colored and multi-spot lights.
- One other second-degree object: cone, hyperboloid, paraboloid..
- Handle bump map textures.

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 Che

Incomplete group Concerning situation Leaks F

Can’t support / explain code

Conclusion

Leave a comment on this evaluation (2048 chars max)

Finish evaluation