mhwangbo

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

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

SCALE FOR PROJECT WOLF3D (/PROJECTS/WOLF3D)

You should correct 1 student in this team

★

Git repository

```
vogsphere@vgs.42.us.org:intra/2018/activities/wolf3d/nobrie
```

Introduction

Please respect the following rules:

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

- Identify with the person (or the group) graded 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 correcting
and the corrected students have reviewed the possible scripts used
to facilitate the grading.

- If the correcting 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.

- The whole group has to be present.

Attachments

Subject (https://cdn.intra.42.fr/pdf/pdf/440/wolf3d.ro.pdf)

Subject (https://cdn.intra.42.fr/pdf/pdf/882/wolf3d.en.pdf)

Subject (https://cdn.intra.42.fr/pdf/pdf/24/wolf3d.fr.pdf)

Play with me too ! (Sierra binary) (/uploads/document/document/407/paris_E3.tgz)

Play with me ! (/uploads/document/document/149/demo.tgz)

Preliminaries

Reminder : Remember that for the duration of the defence, no segfault, nor other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag. This rule is active thoughout the whole defence.

Author file

Check that the author file is at the root of the repository
and formatted as explained in the subject. If not defence is
over and the final grade will be 0.

✓ Yes ✗ No

Display's technical component

We're going to evaluate display's technical elements. Run the program and 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.

- A windows 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 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 arrow keys (we'll accept WASD or ZQSD keys) in the order of your liking. Each keypress must render a visible result on the window, such as a player's movement. If the movement isn't showing, we'll accept a basic display of the keypress in the terminal showing that the keypress event is managed.

✓ Yes ✕ No

Movements

In this section we'll evaluate player's movement inside the maze implementation. 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.

- Press the left arrow (or A or Q) then the right arrow
(or D). The player's view must rotate to the left
then to the right as if the player's head was moving.

- Press the up arrow (or W or Z) then the down arrow (or S).
The player's view must go forward and then backward 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

- The walls 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
clearly visible and correct.

✓ Yes ✗ No

Error management

In this section, we'll evaluate the the program's error management
and reliability. 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.

- 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 alternate or create
unhandled errors.

- Check that there is no memory leaks. You can use the
"top` `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.

✓ Yes ✕ No

Bonus

Reminder : Remember that for the duration of the defence, no segfault, nor other unexpected, premature, uncontrolled or unexpected termination of the program, else the final grade is 0. Use the appropriate flag. This rule is active thoughout the whole defence. 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.

When i'll be older i'll be John Carmack

To deserve being considered as such, a bonus must be:
- Useful (you remain the judge), no need to exaggerate.
- Well done.
- Operational, it cannot generate errors.

Possible examples (1 point per bonus):
- It's not possible to go through the walls (clipping).
- A texture for the ground.
- A texture for the ceiling.
- Objects.
- Doors and keys.
- Monsters.
- A maze made of un-squared walls.
- A skybox.
- etc...

Full marks will be awarded here when implementing either a graphic engine similar to Doom (Walls with random angles and levels on several floors) or an engine similar to Duke Nukem 3D (movements and 3 dimensions free sight). We also recommand you to apply at Id Software and contact Olivier Crouzet (ol@42.fr) to show him your work.

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 · No author file · Invalid compilation · Norme

Cheat · Crash · Incomplete group · Leaks

Conclusion

Leave a comment on this evaluation

Finish evaluation