CMSI 371-01

COMPUTER GRAPHICS

Spring 2013

Assignment 0502 Feedback

Carlos Agudo

2c — You have successfully integrated diffuse lighting computations from the sample code into your scene. This was done correctly; ideally the specular calculation should be integrated also. (|)

2d — You were in class to hear me talk about clipping and hidden surface removal. Yay! (+)

3e — Your shader code, on its own, correctly implements diffuse lighting (see *4a* for where your current bugs live). In addition, specular lighting would expand your fragment shader beyond the trivial "set color" version, so it would be nice for you to get a feel for that also. (|)

4a — Your [lighting] code is overall functional and correct, and fulfills the baseline functionality expected for this course. Your code is broken, though, with regard to the data that get sent to the shader. Specifically, some of your scene objects are setting normal vectors incorrectly (or are missing them completely). Note that normal vectors are crucial for lighting calculations.

As previously discussed, your code has other bugs as well, but this proficiency applies solely to your lighting code and the incomplete definition/handling of normals. (/)

4b — Separation of concerns remains decent and unaffected by your bugs. (+)

4c — Code readability remains the same as before. (+)

4d — For this go-round, you successfully integrated diffuse lighting from the sample code based on the information given. Fix your normal vectors and implement specular lighting for a full proficiency with this outcome in this assignment. (|)

4e — Commit messages are consistently detailed and descriptive. (+)

4f — Lighting functionality was submitted 2 days after the assignment due date. (|)

Updated feedback on assorted outcomes based on commits up to May 10:

1c — You've cleaned up your object composition code a good amount, but have one small thing and one big thing left. The small thing is that you are still calling `passToWebGL` inside `drawObject`, and *this should not be necessary*. The reason you find it necessary is because your recursive implementation still has some gaps.

The big thing is that your instance transform "inheritance" functionality does not propagate the parent transform correctly. It is correct for your particular scene, but not for every possible scene. To illustrate, I've changed your scene code so that one of the child objects on the X-wing has a `trans` object of its own. It disappears just for having that! This is a major gap. (/)

2a — Your raw transform functionality is complete now; this is considered separately from its application, for which recursive handling (*1c*) is not done correctly. (+)

3a — Your interactive functionality is still entirely my code. Yes, you built some keyframes to put together a scene, and yes, you have put in some fixes, but in proportion to what I wrote it isn't on the same level (keyframes does not involve programming; the fixes were more due to instance transforms than the tweening). You need to include another form of dynamic/interactive behavior (e.g., camera to keep the ship in view; another ship, but steerable instead of keyframed; simple physics; etc.) to max this out. (|)

3d — You have added a camera matrix function (although did not include unit tests) and cleaned up the duplicate file for your matrix library. The lack of unit test is big—it should be *automatic for you* that adding a function to a library includes adding a unit test. That keeps this from maxxing out. (|)

4a — Your scene largely works as intended now, but as mentioned it has a number of serious hidden flaws, hidden only because your particular scene does not expose them. So again, it is better but not super. (|)