CMSI 371-01
COMPUTER GRAPHICS
Spring 2015

Assignment 0326a Feedback

Outcome *3a* does not yet cover the entire graphics library for the course so it has a maximum proficiency of | for now. Similarly, because outcome *3d* for this assignment only concerns the vertex shader, that outcome also has a maximum of | for this assignment.

Joaquin Loustau *joaquinloustau / joaquinloustau@gmail.com*

Visible functionality upon execution looks promising…let's see about the code:

1. I think you will see greater flexibility if you reorganize these so that sphere, cube, and icosahedron return the vertices and indices but stop there. Refactor `toRawLineArray` and `toRawTriangleArray` as `prototype` functions of `Shape`. That way, the `Shape` instances do not need to commit to a representation until they really have to. And further, they would be able to generate alternative representations without having to compute vertices all over again. I think the desirability of this flexibility will become more apparent as you use your own framework more extensively. (*4b*)
2. Note how this code, in some form, ultimately belongs to your `Shape` object as a method. (*4b*)
3. Having now seen the `Shape` code, related to note #1 this approach is looking redundant. `Shape.cube` *already returns a* `Shape`, so why not just use that result directly? Yes, some customization is needed, like the injection of the color, transformations, children, etc. But I'm pretty sure that can be coded up pretty quickly. The reward will be much shorter, and potentially easier to read, shape construction code. (*4b*)
4. Decent test coverage here—very reassuring **:)** (*4a*)

1b — +
1c — +
3a (max |) — |
3d (max |) — |
4a — +
4b — | …Some refactoring remains before the `Shape` object fully contains all shape-specific functionality.
4c — +
4d — +
4e — +
4f — + …Consideration given for deadline due to spring break travel.

Updated feedback based on commits up to 2015-05-10:

4b — + …Sufficient design refactoring seen to bring `Shape` to its intended reusability and encapsulation.