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.

Daniel Levine

dannymlevine / dannymlevine@gmail.com

Upon opening *scene.html*, two 404 errors immediately show up; not an encouraging first impression. We'll have to dig in then…

1. A quick-fix change in filename to *scene.js* at least got things going. But this shouldn't have been necessary in the first place—make sure to track your code versions properly! (*4a*, *4b*)
2. Watch your indentation! This is an extremely rudimentary oversight that should no longer be happening at this level. (*4c*)
3. OK, some shapes work seen here, but this is not quite what was intended. The intent here is to have a `Shape` object that *itself* would be the very instance that gets included in the set of objects to draw. As utilized, there is still an "anonymous" object involved. (*1b*, *4b*)
4. There is certainly a library of polygon meshes, including the specifically-requested sphere, so that is good. The double-indirection is puzzling however; I see no need for it. We'll see how this code looks when I get to that file next. (*1b*, *3a*, *4a*)
5. *** Based on the shapes defined in this scene, there is no evidence here of composite shape functionality, another explicitly requested work item in the assignment. (*1c*, *4a*)
6. With a fully-envisioned `Shape` object, this code would be one of the portions that can be refactored as a `prototype` function/method. (*4b*)
7. Function arguments in new lines should be indented. (*4c*)
8. *** Definitely no sign that shapes can handle children yet. (*1c*, *4a*)
9. *** Wait what? Those function calls returned empty objects? This is not right. (*1b*, *3a*, *4a*)
10. OK, so *this* is where the heavy lifting is happening. In terms of raw functionality, yes, you have successfully created a few polygon meshes. In terms of design, however, these are in a very rudimentary form. Everything else listed in the `objectsToDraw` array should be folded into the final `Shape` object that should have been defined. The vertices and their indices just happened to be the beginning. And as mentioned in note #4, the double-call is completely unnecessary. A function like `Shapes.sphere()` should have done everything needed to create that sphere polygon mesh. (*1b*, *4b*)
11. Ummm, nope, `MAGIC_NUMBER` is not a good name for this variable. (*4b*, *4c*)
12. Thank you for crediting the resource used in helping with the sphere. (*4d*)
13. OK, good start…but clearly you have a ways to go here. (*4a*)

1b — | …Thanks to the different defined polyhedra, this at least stays at a |. The missing `Shape` object is what keeps it from going higher.

1c — / …Unfortunately, this functionality is totally absent.

3a (max |) — / …The polyhedra again save this one, but due to the maximum of | it is necessarily constrained because the maximum proficiency cannot be given without a `Shape` object.

3d (max |) — / …The total absence of composite shapes means that no changes were made to the drawing code at all, which was not the intent of this assignment.

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.

4a — / …The library of polyhedra constitutes around half the expected functionality.

4b — | …Nothing egregious seen, but also without the refactoring that would have been triggered by a correctly-designed `Shape` object it cannot be given maximum proficiency.

4c — | …The code presentation is mostly good, but are spoiled by some really avoidable glitches.

4d — | …Good move with crediting *webglacademy.com*, but beyond that you could have inquired more deeply about the very nature of a `Shape` object.

4e — +

4f — +

Updated feedback based on commits up to 2015-04-30 *(no written notification was given on this resubmission, but as a courtesy I will throw these in because I happened to notice them when grading Assignment 0430a and 0430b)***:**

Overall `Shape` object functionality is largely fulfilled, with loose ends remaining in these areas: first, the prebuilt shape objects themselves need some tweaking. The sphere is fine, but the cylinder's mesh needs cleanup (as evidenced by its non-solid appearance when rendered as triangles) and the double pyramid has both some stray triangles as well as triangles specified in the wrong order (apparent when lighting is set).

Next, the overall *design* of `Shape` still needs some work. One area is a lingering disconnect between what the prebuilt shapes return and how they are used. Yes, they now return a `Shape` object…and yet when assembled in the final seen, *they are called twice* and *yet another* `Shape` *object is created*. There are multiple inefficiencies here, as well as a continuing misunderstanding of how the `Shape` object is supposed to represent, in a single package, everything that needs to be known about a particular 3D object in the scene.

The second design gap is the lack of conversion of the many sample-code functions into `prototype` functions for the `Shape` object. This again follows the theme of turning a `Shape` object into a complete, self-contained object that knows how to manage and render its information without the help of outside code.

Based on the revisions, final outcomes here are:

1b — | …The top-level `Shape` object is certainly a step forward, but due to the issues above a good chunk of work remains before justifying an upgrade.

1c — | …Core functionality is there, but should move into the object to get that additional bump.

3a (max |) — | …Some loose ends as mentioned but good enough for a bump.

3d (max |) — | …`Shape` design isn't fully there for child objects, but drawing code is.

4a — | …The additional refactoring is good for one notch up, with the missing gaps needed to top off.

4b — | …This outcome needs to stay put due to the design-oriented nature of the lingering items.

4c — | …No major changes in code presentation; misindented lines are still there, with some lines being excessively long.

4d — | …Definite benefit seen from available course information, but design glitches show a continuing need for some object-oriented programming fundamentals.