Programming assignment #8

Due Wednesday by 11:59pm **Points** 4
Submitting a text entry box or a file upload **Available** until Dec 17 at 11:59pm

Due: Wednesday, December 14th (at midnight).

Late Policy: Assignment submission will remain open for 3 additional calendar days after the formal deadline. Late submissions will be penalized by a maximum of 1 point on our 4-point scale (i.e. if you submit 3 days late and your assignment was otherwise deserving a "3", you might get a grade as low as a "2" due to this penalty). Some leniency might be exercised on this general rule based on how late the assignment was, and whether turning in an assignment was a repeating occurrence or a one-time event.

Synopsis: You get to use WebGL to create a 3-dimensional scene using textures to enrich the visual appearance of your objects.

Learning Objectives: To familiarize yourselves with the complexities of using WebGL to manipulate textures, load texture images onto the GPU, specify texture coordinates for your models, and adjust shaders to leverage texture mapping functionality.

Evaluation: Based on our 4-point grading scheme, as discussed in our introductory lecture. You get a check ("3") if you turn in a viable, and complete submission (even if it just draws a rectangle like the example in the tutorial). "Above and beyond" grades (i.e. a "4") will be awarded for people who have crafted something particularly cool. As a general rule, no more than 1/3 of all assignments turned in (the very best ones, that is) will be considered for a "4" grade.

Collaboration policy: This is an assignment to be done individually. Code not written by you needs to include proper attribution (see **this post (https://canvas.wisc.edu/courses/320922/pages/collaboration-policy)** here). It is always ok to use code provided in our in-class examples as a starting point, but you need to add your own effort to raise those examples (or other sources) to what is asked by the programming assignment (i.e. finding some code on some online forum that does all the job for you that is needed to satisfy the assignment is not the intent, if you haven't added any of your own effort to it). If you use somebody else's code (other than our GitHub examples), make sure to clarify in your submission notes what **you** did, and what you repurposed from the external source.

Hand-in: Electronic turn-in on Canvas. Make sure that you turn in all files needed for your program to run. It is acceptable to turn in a single HTML file with your program, but even preferable to separate your code into an .html file and a separate .js file containing the JavaScript code, similar to the examples in our **GitHub repository**

(https://github.com/sifakis/CS559F22_Demos) (see, e.g. Demos in Week10/ Week11/). If you submit anything else than a single HTML file, please put everything in a single ZIP archive. Feel free to use the copy of the glMatrix library included in our examples in the GitHub repository (or use them as a starting point) if it's convenient. ***It is not acceptable to submit a link to JSbin for this assignment!***

Description

Your task will be to create a 3D scene, visualized using the WebGL drawing API, very similar to what was requested of you for Programming Assignment #7. As before it will be your own responsibility to write your own vertex/fragment shader pair(s), compile/link them into a "program", define vertex attributes, send them (via buffer objects) to the GPU, and establish the necessary transforms (lookAt, projection, etc) as "uniforms" that are dispatched to the GPU.

The difference between this assignment and the previous one (#7) is that this week you will be required to use texture mapping somewhere in your scene. Specifically, you should satisfy the following requirements (some of which overlap with those of assignment #7):

- Your scene should include at least one "polyhedral" object with multiple shaded (as opposed to be drawn as "wireframe", only by their edges) polygonal facets. Those will be typically be comprised of triangles. Your entire object cannot be all flat! (unless if you include several objects in your world, in which case it's ok for at least one of them to not be flat). We would like to be able to visually appreciate that the Z-buffer visibility algorithm is actually working … "front facing" triangles/polygons, should hide parts of the objects that are located behind them.
- You should use at least one texture somewhere in your scene. It is perfectly fine to re-use some of the textures that are included in the in-class examples (see the JSBin demos or the ones in the GitHub repository from Week11). In our in-class lecture, we discussed (verbally, for those that were present) the process for serving an image texture through Flickr (these (https://live.staticflickr.com/5564/30725680942_0c6e60a13f_o.jpg) are (https://live.staticflickr.com/65535/50641871583_78566f4fbb_o.jpg) some (https://live.staticflickr.com/5323/30998511026_c90053af9c_o.jpg) of (https://live.staticflickr.com/5726/30206830053_87e9530b48_b.jpg) the (https://live.staticflickr.com/65535/50642695166_9c28ba57e8_o.jpg) examples (https://live.staticflickr.com/65535/50641908943_f6ebfef28d_o.jpg) from your instructor's Flickr collection, which area ready-to-use for you if you wish - those links are already of the right "type and size" - or you can upload and use your own). Expect to see a piazza posting soon with a verbal review of the process.
- You should have a texture coordinate vertex attribute for your model (or whichever model in your scene actually uses a texture), and correspondingly Sampler2D variables in your shaders.
- Texture look-ups should occur in the fragment shader (as in our in-class examples), not in the vertex shader.

- At least one such texture should "wrap around" multiple triangles/polygons in your object. For example, having the cube model from our in-class examples where each square face of the cube has a complete copy of the texture image would not be appropriate; instead consider if the texture image was stretched/wrapped around 4 square faces surrounding the cube.
- You should have a textured model (or at least one of your models) that's substantially different from the cube model shown in class. A "minimally acceptable" possibility could be for example a "house" model, with a cube as its main part, and a square-based pyramid as its roof (it would be nice in such case to have a texture wrapping around the 4 triangles that make up the roof!).
- You are not required to use MipMapping, but it is recommended.

The simplest way that you can use this texture to create a visual appearance is to use the color resulting from the texture look-up as the "diffuse color" of the object, as in our early examples **[JSBin]** (https://jsbin.com/zivofiw) . ***Note that, if you already used textures in your programming assignment #7, it could very well be the case that your previous submission already satisfies the requirements of assignment #8 as well! In this case (if you used textures that satisfy the above requirements) you can simply resubmit the same, and obtain a "3" for this assignment as well!*** But if you want to be more competitive and aim for a higher grade, consider being more ambitious along the following lines:

- Use multiple textures, and multiple objects with interesting shapes.
- Use more "advanced" texture mapping effects. Decal textures and normal mapping would probably be the easiest ones to attempt. More advanced effects that use render-to-texture would also be very welcome, but please consider that those can be significantly more difficult (but success will be rewarded).