Programming Assignment #3

Due Oct 22 by 11:59pm **Points** 4 **Submitting** a text entry box or a file upload
Available until Oct 27 at 11:59pm

This assignment was locked Oct 27 at 11:59pm.

Due: Friday, October 22nd (at midnight).

Late Policy: Assignment submission will remain open for 5 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 5 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 will make a program with an object (or more) that uses the concept of hierarchical modeling and have it be animated, using explicit matrix and vector representations for the points and transforms involved.

Learning Objectives: To obtain experience in using mathematical representations of 2D transforms (i.e. matrices), how such transforms are applied to homogeneous representations of vectors, and how transforms are combined.

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://graphics.cs.wisc.edu/WP/cs559-fall2020/2020/09/10/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/CS559F21_Demos)** (see, e.g. Demos

0-2 from Week 5). If you submit anything else than a single HTML file, please put everything in a single ZIP archive. ***It is not acceptable to submit a link to JSbin for this assignment!***

Description

In your previous programming assignment, you were asked to create a scene with one of more objects that are created according to the principles of hierarchical modeling. In creating such objects, you were asked to leverage the HTML5 Canvas transformation commands (i.e. the **translate(), scale(), rotate()** methods of the drawing context) and the stack supported within canvas (manipulated via the **save(), restore()** methods) to create models with multiple components whose transforms hierarchically depend on each other.

This week's assignment asks you to produce something that *visually* could be very similar or even identical to the result of Assignment #2; however, instead of using the Canvas transforms and stack, you are asked to implement any transforms (and, by "implement", we mean creating all the elementary transforms, combining them to create composite transforms, applying them to point locations to convert them from one coordinate system to another) using explicit matrix representations, using a matrix/vector library such as **glMatrix** (http://glmatrix.net/) or **TWGL** (https://twgljs.org/) that we discussed in class. Although glMatrix would be gently recommended, since we had the opportunity to cover it in more detail in class, you are absolutely welcome to use TWGL or some other matrix library instead.

One possible idea, if you are happy with the result you turned in for Programming Assignment #2 would be to first create a version of the same product, that is visually identical to what you implemented using the Canvas transformation functionality, but internally substitute all the calls to the **translate(), scale(), rotate()** calls (or the **save(), restore()** methods for the built-in transform stack) with glMatrix transform representations and operations. We saw a very good example of this in class; we examined a hierarchically modeled object implemented via Canvas calls **[JSbin link]** (https://jsbin.com/hozeyar), and then saw how the exact same functionality (and appearance) can be implemented using glMatrix representations of transformation matrices **[JSBin link]** (http://jsbin.com/yifahoq). We also saw an example that uses TWGL as the underlying matrix library **[JSBin link]** (https://jsbin.com/qutelof). The only canvas transform-related operation you are allowed to use (it's only optional to do so …) is the setTransform() method, if you don't end up applying the transform yourself (like in this example **[JSBin link]** (http://jsbin.com/leropag)). You can also optionally implement your own stack (you are not allowed to use Canvas' own stack, as in the **save(), restore()** methods) as we saw in examples in-class **[JSBin link]** (http://jsbin.com/zimaqos). You can find implementations for all these examples in our **GitHub repository** (https://github.com/sifakis/CS559F21_Demos) under directories **Week4/** and **Week5/**.

Although it is perfectly acceptable to produce a result that is identical to your programming assignment #2, as always we encourage you to be creative. If you want your submission to be competitive for a "4" score, similar embellishments that would boost your score in the previous assignment (in terms of aesthetics, complexity of motion, intricacy of model) would also be considered here; however note that if you previously got a "4" for being creative, but now you turned

in the exact same visual result, it might not be enough to *just* change the matrix representation of the transforms to get a “4” again … you will need to be creative in a similar but different way.

One possible idea in your quest for making your assignment more interesting: You may consider using some simple parametric curves like the ones we have been recently talking about, to either add interesting shapes, or create an interesting motion/animation by making certain objects move along a curved trajectory ... just an idea, certainly not a requirement!