CSE 470 Homework #1: T W I S T Y !

Big Picture:

Design an object (or something artistic) with triangles. Animate the design with a ``twisty" action that is a variable rotation, causing a deformation of the design. The design should rotate counterclockwise and then reverse direction and rotate back to the original position. Some part of the design must remain stationary at all times. As the design reaches the end of the counterclockwise rotation, it fades to the background color. At the end of the clockwise rotation the original colors will be restored. This action is repeated as long as the program is running. See the specifications section for exact requirements and definitions.

Learning Objectives:

1. Understand and modify an existing webGL program.
2. Understand the structure and interactions between HTML, JS, webGL, GLSL.
3. Create a geometric object defined by triangles.
4. Learn how to create an animated graphics program.
5. Set color based on the dynamics of the program.
6. Learn to send data and other variables to the vertex shader.
7. Experiment with transformations in the vertex shader.
8. Create effective debugging code and comments.

Specifications:

1. Start with twisty.html and twisty.js files. You are required to add *both code and code comments* in the locations marked with "HW470" in the HTML, GLSL, and JS codes.
2. Create an interesting object or artistic design. Points will be rewarded for creativity.
    a. The object must have at least 30 vertices. (Note: the same 2d coordinate location may appear more than once in the vertex list and count more than once towards the 30.)
    b. The object must be defined by solid-colored triangles. (This means that each vertex of a triangle is assigned the same color.)
    c. In total, at least 5 colors must be used.
3. Animate a majority of the design so that it rotates *d* degrees counterclockwise and then reverses direction and rotates back *d* degrees. The angle *d* should be 90 degrees or greater.
    a. In addition to rotation, the object will undergo a deformation by calculating the rotation angle based on the distance of a vertex from the origin. (This calculation occurs in the vertex shader.)
    b. The initial geometry and the entirety of the rotation must be contained inside the window.
    c. Some part of the object must remain stationary.
    d. The modification of the vertices for the rotation must take place in the vertex shader.
4. Make the color of the moving parts fade to the background color (white) as they rotate.
    a. Linear interpolation is the simplest/best option.

   b. The stationary part’s color must remain constant.
   c. Creation of the fading/unfading color must occur in the vertex shader.

Tips:

1. Get started early.
2. Get help early if you get stuck at any stage. (Come to class for more tips!)
3. Make sure you understand simpleSquare, rotatingSquare, and review the class notes.
4. See the Programming Tips list.
5. Make a plan of action before writing any code.
   a. Recommended order: create geometry, display static image, animate, color change
6. Work on one task at a time and thoroughly test before moving to next task.
7. PRINT debugging code! Avoid the silent program. (Unfortunately printing to the “console” in the shaders is not possible. We will discuss options in class.)
8. Make a paper sketch of your object and label the vertices before coding them.
   a. Note: You can build the object with parametric equations (e.g., a circle).

Submission Instructions:

1. Turn in your assignment on Canvas.
2. All assets you created must be in one zipped file. Do not include the Common folder. The references to the files in the Common folder must be as done in the class examples.
3. Name your zip file: LastnameFirstInitial_HW1.zip.
4. We will grade the last submission only.
5. Review the late submission policy in the Syllabus.