CS 371 – Assignment 2
Phenomenal Fractal Fantasies Mightily Morphed!
Due: By 11:59pm, Monday, October 2nd

"So, naturalists observe, a flea
Hath smaller fleas that on him prey;
And these have smaller still to bite 'em;
And so proceed ad infinitum"
Jonathan Swift - On Poetry, A Rhapsody

This assignment will give you a chance to tie together many of the techniques we have been discussing in class - two-dimensional transformations, fractals represented in IFS form, the fractal spray-paint algorithm, morphing, and minimal user interaction. The only required interactive input I will provide as I run your program is to strike the "r" key to reverse run a completed animation. You are certainly free to add more as you strive for GGW points.

The basic premise of the program that you will develop is that you will load two IFS objects (recall how the fractal example discussed in class used just one such object). After you have read two IFS objects, your JavaScript CPU program should compute an appropriate number of vertices for each of them.

Then your shader code should begin to do its job as it morphs one fractal into the other. See `http://csf11.acs.uwosh.edu/fractal_fantasies.gif` for what such a morph might resemble when the two fractals involved are a fern-like fractal and the famous *Koch fractal*. The essence of such a morph is incorporated in the linear tweening transformation we discussed in class, i.e.,

$$tween(t) = start \times (1-t) + goal \times t$$

.

Here are the minimal requirements that your program must meet for the 90%-level.

- When I view your animation, I should see a smoothly running morph of one fractal into the other. When the morph is completed, I should be able to view the resulting fractal until I strike the "r" key. When I strike "r", it means I want to run the morph in the opposite direction, after which I'll again sit staring into the self-similar depths of the fractal until I strike the "r" key again.
- Near the top of your HTML page, I should see the (correctly updated) message:
  `MORPHING XXX TO YYY`
  where XXX and YYY are the names of the start and goal fractals (don't forget to reverse them when appropriate). For your pleasure in experimenting as you develop the program, there are a collection of IFS objects available via a link from D2L. These objects are not yet assigned to a JS variable as was the example we covered in class. This is because you will need to use at least two such variables, so you will have to edit the files to assign them to the variable names you choose to use in your program.
- Not only should the fractal shapes themselves be morphed, but some morphing of their colors should occur as one fractal's shape takes on the shape of the other.
- Throughout my viewing the morph, the image display should always take up nearly the entire canvas. Since different fractals need different WC windows to ensure this, you will no doubt need to adjust the WC window on the fly as the morphing animation happens.

What should you submit?

In one zip file that you upload to the D2L dropbox for Assignment 2, I should find:

- One HTML file named *assignment2.html*
- One JavaScript file named *assignment2.js* with your window.onload and render functions.
- Two files that contain the JSON representation of the fractals you have chosen.

To load everything correctly, the following must be part of your HTML file:

```
<!-- The A/S WebGL support libraries-->
<script type="text/javascript" src="../Common/webgl-utils.js"></script>
<script type="text/javascript" src="../Common/initShaders.js"></script>
<script type="text/javascript" src="../Common/MV.js"></script>
<script type="text/javascript" src="../Common/webgl-debug.js"></script>
<script type="text/javascript" src="XXX.js"></script>
<script type="text/javascript" src="YYY.js"></script>
<script type="text/javascript" src="assignment2.js"></script>
```

where XXX and YYY are the names of the files containing the JSON IFS representation of your two fractals. If I have to change any of this in order to run your program (for instance, because you didn't follow our conventions as to where the A/S Common utilities are located), a 10% penalty will be imposed.

As always, any GGW efforts should be documented to ensure you receive credit for them. That documentation can appear in the introductory documentation block of your *assignment2.js* file or, even better, add it directly into your HTML page so that viewers of your efforts are appropriately informed regarding all the neat stuff you have done.