Resources

- WebGL
- Wrote code in *Sublime Text* editor.
- Used Javascript tutorial from site http://www.w3schools.com/js/.
- Used code from site http://learningwebgl.com/blog/ as template, including .js files required for main program to run. This code in turn was originally based on the NeHe tutorials from http://nehe.gamedev.net/. Also used both of these sites as a tutorial for webGL.
- Used Javascript code style guide from site https://code.google.com/p/google-styleguide/.

Platform Problems

- I do not have a comp sci background, so it was challenging to figure out which software to use and how.
- There are many online tutorials on graphics programming, but I did not know which to use, and would spend time beginning one before realizing it was not useful.
- Originally tried to use XCode and OpenGL, but settled on WebGL because downloading XCode requires an app developer profile on the Mac App store, which costs money.
- Most online support directed toward people with stronger programming background.

Programming Problems

- No previous Javascript experience, so had to acquire working understanding quickly.
- WebGL does not have a way to let programmer know the location in the script at which an error has occurred. Makes it more difficult to identify sources of error.
- Aside from that, no real issues other than standard errors that needed de-bugging.

Design Choices

- Included variable parameters in sinusoidal scaling functions, which I made different from those suggested in the assignment. The reason for this was to make the scaling slighter, so that the mechanism was clear to the viewer. If scaling becomes too great, it can appear as though rectangles are rotating with respect to x or y axes.
- Chose condition for wrapping around based on the wish to not have any rectangles disappear while still visible. Thus, based on the dimensions of the screen, the maximum possible length and width, and the maximum possible instantaneous scaling, I calculated the maximum possible distance that the centre of a rectangle could be from the origin (centre of the screen) and still be visible. I specified in my code that if any rectangle reaches such a point, it is guaranteed that it is fully off the screen, and it will wrap around to the other side. Thus, most rectangles spend several second off-screen before reappearing on the other side, but none disappear while still in view.
- All other parameters such as range of size, range of speed, etc., were chosen in order to make all characteristics of the group of rectangles' movement clear. For example, I chose the size range large enough that we can see rotation and scaling clearly, but small enough that we can see many different rectangles at once.