CS 4053/5053
Homework #2 – Drawing in OpenGL
Due Tuesday 2018.02.20 at the beginning of class.

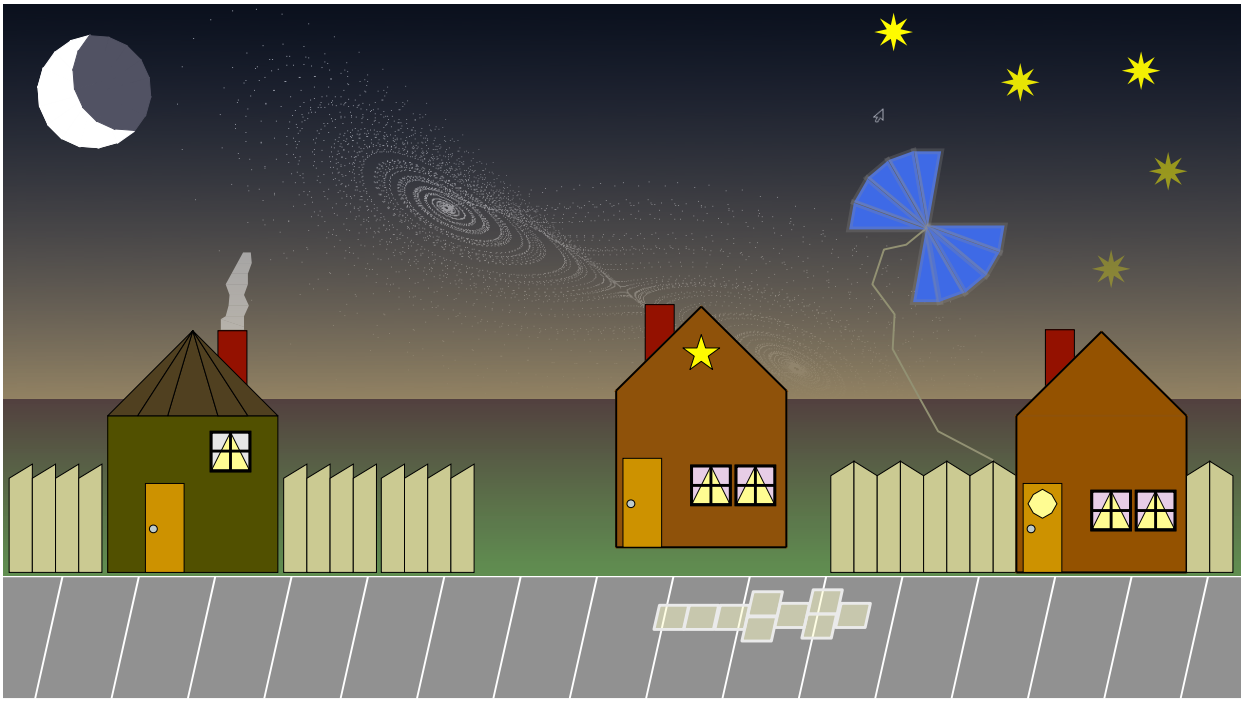
All homework assignments are individual efforts, and must be completed entirely on your own.

In this assignment you will learn how to draw scenes using the JOGL OpenGL graphics library. Specifically, you will use the drawing primitives and rendering options in OpenGL to recreate a picture drawn by hand in presentation software (Keynote '09 for Mac OS X).

Consider the picture at the bottom of this page. It contains a wide variety of graphical objects made up of points, lines, and polygons. Many of the polygons are triangles or quadrilaterals. They are rendered using a mix of solid coloring, translucency, and gradient fills. Some of the lines and edges are thicker than others. These features were all created, positioned, and styled using common drawing features of the presentation software, with the exception of the galaxy in the background.

This assignment has three parts. The **first part** to learn how to build your homework apps using Gradle. Spend some time at https://docs.gradle.org/current/userguide/userguide.html to get a feel for what Gradle does and how it works. You won't have to write your own Gradle scripts to compile and run your code from a command line; I've done that for you. You will have to install Gradle on your system to use it. It's open source, so you can get it in the usual ways like downloading directly, compiling the source, using a package manager, etc. (Note that if you want to use run and compile your code from inside an IDE like Eclipse, you'll have to figure that out for yourself for your particular environment.)

Once you have Gradle installed, go into the `ou-cs-cg` directory that came with this homework. The directory contains a `build.gradle` script file. Have a look at it; there's a summary of build commands at the top. (Also make sure to look at the section on "Alternative Start Scripts". You'll



need to add to it to build future homeworks.) To compile the build, type `gradle installDist` on the command line. You need to be in the `ou-cs-cg` directory for this to work. Then go into the `build/install/template/bin` directory to run any of the resulting programs. Each app has a `.bat` version for running on Windows. The `template` program is very basic, but you can use the corresponding class in `Template.java` as a starting point for your own programs. You can find the source code deep in the `src` directory. Your own code should go in the `homework` directory, and your classes should be in the package `edu.ou.cs.cg.homework` (or a subpackage). I've copied `Template.java` to `Homework02.java` (with corrected class and package names) to make it easier for you this time.

The **second part** is to study how an example program can be written using JOGL. The `gasket` program animates a multihued triangle overlaid with a rendering of the Serpinski gasket. It's the one I showed in class. The corresponding code is in the `Gasket.java` class. Study the code and how it's organized in this class to learn more about how JOGL renders and can animate an OpenGL canvas inside a Swing UI. Keep the JOGL API (http://jogamp.org/deployment/jogamp-next/javadoc/jogl/javadoc/) handy—specifically the `com.jogamp.opengl` package—to look up details about particular classes and methods. Also spend some time browsing the API to learn more about what's available. Start with the `GL`, `GL2`, and `GL2ES3` classes.

The **third part** is to reproduce the picture on the previous page using JOGL. Write your code in the `Homework02.java` class to generate the scene. Add and call additional private methods as you like to help modularize your code. You may create additional classes if it's appropriate. **Document all classes, objects, methods and steps in your code thoroughly.**

Everything in the picture can be reproduced using the OpenGL drawing functions that we've already talked about in class, including the galaxy in the background. The galaxy can be drawn using only points. The tricky part is figuring out which points to draw. The Lorenz equations (https://en.wikipedia.org/wiki/Lorenz_system) produces trajectories that looks a lot like a galaxy. Although such differential equations often require careful numerical solution (such as integration using a high-order Runge-Kutta method), simple iteration and addition works for our purposes. You can find some helpful code to do that at http://www.algosome.com/articles/lorenz-attractor-programming-code.html. You'll have to figure out how to map the 3-D coordinates of Lorenz points into 2-D coordinates to reproduce the picture. If you've tried your best but haven't come close, feel free to map the coordinates in another way that recreates the look of a galaxy. The Tinkerbell map (https://en.wikipedia.org/wiki/Tinkerbell_map) is another alternative worth trying.

You will be graded on: (1) how completely and accurately you reproduce the picture, and (2) the clarity and appropriateness of your code. **Don't go to extremes trying to achieve a pixel-perfect reproduction!** Moderate variation between the picture and scene is to be expected. Nonetheless, all of the graphical elements and effects can be reproduced using OpenGL.

Bonus points may be awarded for using an exceptionally clever method to efficiently reproduce: (1) all shape and color details of the moon; (2) the galaxy look using only points.

To **turn in** your homework, first append your 4x4 to the `homework02` directory; mine would be `homework02-weav8417`. Second, take a screenshot of your scene, trim it, and put it in the `Results` subdirectory as `snapshot.jpg` or `snapshot.png`. Third, run `gradle clean` to reduce the size of your build before turning in your homework. Leave your java files where they are in the build. Zip your entire renamed `homework02` directory and submit it to the Homework 02 assignment in Canvas.