CS 4053/5053
Homework #3 – Interaction in JOGL
Due Tuesday 2018.03.06 at the beginning of class.

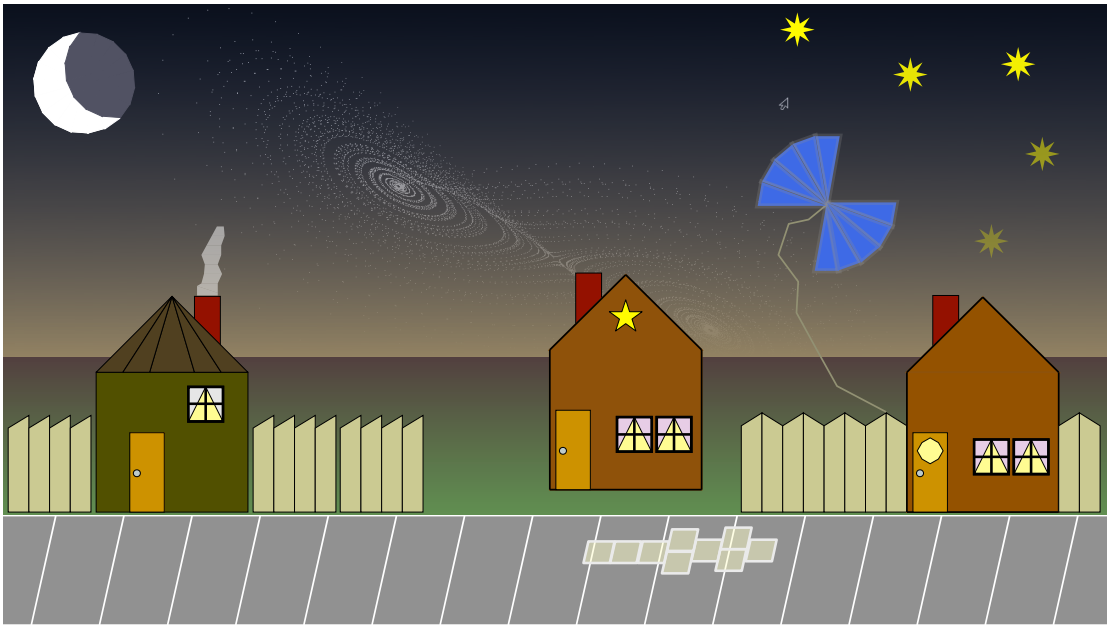
All homework assignments are individual efforts, and must be completed entirely on your own.

In this assignment you will learn how to support interaction using the JOGL OpenGL graphics library. Specifically, you will use the mouse and keyboard interaction capabilities of Java to let the user create and manipulate elements of a scene rendered in OpenGL.

In homework #2, your recreated the picture at the bottom of this page using OpenGL. In this homework, you will add interactions that allow the viewer to alter the scene. You will provide a set of mouse interactions to move objects and keyboard interactions to change how they look.

This assignment has three parts. The **first part** is to update your Gradle build. Start with the `ou-cs-cg` directory distributed with this assignment. The `build.gradle` script has several new `createScript()` lines to create the executables that we have encountered up to this point. (If for some reason you changed `build.gradle` for homework #2, update the new one correspondingly.) In `src`, merge in any code that you wrote for homework #2, including your revised `Homework02.java`. I have included my own version, `Homework02Chris.java`, that you may start from for this assignment if you prefer.

The **second part** is to study an example of using JOGL to combine Java interaction handling with OpenGL rendering. Run the `interaction` executable. Press arrow keys to translate the origin, click and drag the mouse to draw a polyline, or press `'delete'` to delete the polyline. The corresponding code is in the `ou.cs.cg.interaction` package. Study the code and how it's organized to learn more about how you can: (1) model the state of a view using a set of Java objects; (2) process Java events to change those objects; and (3) tell JOGL to re-render the scene when the model changes. Keep the Java API (http://docs.oracle.com/javase/8/docs/api/) handy—specifically the `java.awt.event` package—to look up details of particular classes and methods. Spend some time browsing the API to learn more about what's available. Start with the `KeyEvent`, `MouseEvent`, `KeyListener`, and `MouseListener` classes.



The **third part** is to add a variety of fun interaction features to your scene rendering code. Some of them will require you to modify your rendering code. Make a copy of `Homework02.java` (yours or mine), call it `Homework03.java`, and add your code to that. Alternatively, make a copy of the entire `interaction` package, call it `homework03`, and copy the rendering code from `Homework02.java` (yours or mine) into the new `View.java`. Either way, uncomment the corresponding `createScript()` line in `build.gradle` to create the `hw3` executable. Use methods and inner classes to modularize your code. Feel free to create additional classes if it's appropriate. **Document all classes, objects, methods and steps in your code thoroughly.**

Add the following interaction features:

- The `<page up>` and `<page down>` keys increase and decrease the height of the fences.
- The `<w>` key toggles whether the window shades are opened or closed.
- The `<1>` to `<9>` number keys change the number of panels on each side of the kite, but without changing the total angle (about 100 degrees) for each half.
- The `<left arrow>` and `<right arrow>` keys move the hopscotch squares left and right by `1.0` sidewalk slab if <shift> is down, `0.1` sidewalk slab otherwise.
- The `<up arrow>` and `<down arrow>` keys move the hopscotch squares toward and away from the houses by `0.1` sidewalk slab, but always staying inside the sidewalk.
- The `<tab>` key selects one of the stars. Starting with no star selected, `<tab>` cycles through the stars one at a time, then returns to no star selected. The selected star is filled in orange instead of yellow.
- A mouse `<click>` moves the selected star to the click point, but only if the click point is above the horizon.
- A mouse `<drag>` draws a polyline from the top of the fence to the center of the kite. Mouse dragging events proceed like this: `<press>`, `<drag>`, ..., `<drag>`, `<release>`. When there's a `<press>`, start a new polyline from the top of the fence to the press point. For each subsequent `<drag>`, add a point to the end of the polyline. Draw the kite centered at the final point in the line. Draw the kite translucently during a drag and normally afterwards.
- **BONUS**: Without disrupting any of the interactions above, include additional interactions that let the user: (1) add a randomly positioned star, (2) delete the selected star, and (3) `<drag>` the *center* of all stars to a new point. All stars must be in the sky at all times!

You will be graded on: (1) how completely and usefully you support the above interactions, and (2) the clarity and appropriateness of your code. All of the interactions can be reproduced using Java events and JOGL. Don't go to extremes trying to get interactions and graphical effects exactly right. Variation in behavior and rendering appearance is expected...and entertaining!

To **turn in** your homework, first append your 4x4 to the `homework03` directory; mine would be `homework03-weav8417`. Second, perform several interactions to change the objects in the scene, take a screenshot, trim it, and put it in the `Results` subdirectory as `snapshot.jpg` or `snapshot.png`. Third, write a brief description of the interactions that you performed and what your code did (generally) to create the screenshot. Put it in the `Results` subdirectory as `caption.txt`. Fourth, 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 `homework03` directory and submit it to the "Homework 03" assignment in Canvas.