CS 4053/5053
Homework #4 – Vectors Victorious
Due Tuesday 2018.04.05 at the beginning of class.

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

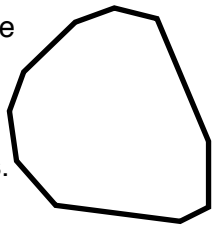
In this assignment you will learn how to animate graphics with basic collision dynamics using vector calculations and the JOGL OpenGL graphics library. Specifically, you will calculate the frame-by-frame position of shapes moving around in an enclosed space.

The parts of the assignment are designed to be done in order, with new features being added in each part. I don't expect the majority of you to complete all the parts! I will grade progressively and be generous with partial credit. It's all doable with the vector math covered in class and the textbook. For each part, carefully think through how to model objects and movements as points and vectors. **I strongly urge you to review sections 4.1–4.6 of the textbook before starting.**

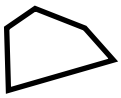
This assignment has five parts. The **first part** is to update your Gradle build. In homework #3, I provided a basic JOGL application framework in the `interaction` package. You're welcome to use your own code, but I recommend starting with my code since you won't be building on homework #3. If you want to preserve your old code, feel free to make a new package and copy the code you need into it. Don't forget to change the package names in the class files if you do that. Update the line in `build.gradle` to build the `hw04` application target correspondingly.

The **second part** is to implement an animation (use 60 FPS) of a single point moving around inside a box. Inset the box by about 10% from the edges of the window. Give the point a random direction and a starting speed that takes about two seconds to cross the box. In each animation frame, update the position of the point, reflecting it off of the walls of the box to keep it inside. Let the user hit the `<left arrow>` and `<right arrow>` keys to slow it down by 0.9x or speed it up by 1.1x, respectively. Don't put limits on minimum or maximum speed.

The **third part** is to let the container be some other convex polygon. Define three more **large** polygons for the point to bounce around in: a regular hexagon, a regular 32-gon circle, and a polygon that looks more or less like the one at right. Make them as large as you can within the same window insets as the box. Let the user dynamically pick which polygon by hitting the `<1>` to `<4>` number keys. You may teleport the point to the center whenever the polygon changes.



The **fourth part** is to let the moving object have a shape. Define three **small** convex polygons as alternatives: a square, a regular octagon, and a polygon that looks more or less like the one at right. Size them so that their area is about 0.1% of the area of the container polygons. Let the user dynamically pick which shape (the point or one of the three polygons) by hitting the `<6>` to `<9>` number keys. Let the user hit the `<down arrow>` and `<up arrow>` keys to decrease the area of the object by 0.9x or increase it by 1.1x, respectively. (When the moving object is a point, don't change it's size but do remember any size changes, and use the current size whenever switching objects.) Don't put limits on the minimum or maximum area. *How does a sudden area change affect the reflection calculation?*



The **fifth part** is to let the user click inside the container polygon to create additional shape polygons. Center each new one at the click point. How should you adjust if this causes the shape polygon to fall partially outside the container polygon? (Note: The shape polygons DO NOT need to interact with each other, just the walls of the container polygon.)

There are several bonuses available, with a maximum score of 20/20. The bonuses are also designed to be independent but increasing in difficulty. The first three are intentionally relatively easy to help you if you have a lot of difficulty on the later parts above.

BONUS: Give each side of the container polygons a color. After each bounce, set or modify the color of the shape polygon according to the side color.

BONUS: Give each side of the container polygons an "impulse factor" other than 1.0. After each bounce, scale the magnitude of the shape polygon's movement vector by this factor.

BONUS: Have each shape polygon except the original one fade a little on each bounce, and disappear entirely after 20 bounces.

BONUS: In several cases above, you can simplify things by moving each shape polygon to the center of the container polygon whenever the user interacts to change something. This isn't very smooth or natural looking, however. Devise and implement an effective strategy to keep shape movements smooth and natural looking when: (1) the container polygon changes; (2) the shape polygon changes; (3) the shape area changes.

BONUS: Have all shape polygons bounce off of each other as well as the walls.

You will be graded on: (1) how completely and correctly you realize movement and interactive user changes in the above parts, and (2) the clarity and appropriateness of your code. Don't go to extremes trying to achieve perfect graphical effects. Variation is expected, and getting it "good enough" is the hallmark of graphics programming!

To **turn in** your homework, first append your 4x4 to the `homework04` directory; mine would be `homework04-weav8417`. Second, write brief descriptions of the procedures that you followed to achieve each of the five parts and any of the bonuses that you attempted. Incorporate these descriptions as longer blocks of documentation in your code (and label them to make it clear that that's what they are). 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 `homework04` directory and submit it to the "Homework 04" assignment in Canvas.