L7: Program Structure

Playing with magnets

The Scenario. We would like you to write a program that simulates the action of two bar magnets. Each magnet will be represented by a simple rectangle with one end labeled "N" for north and the other labeled "S" for south. Your program should allow the person using it to move either magnet around the screen by dragging it with the mouse. You know that opposite poles attract, while similar poles repel each other. So, if one magnet is dragged to a position where one or both of its poles is close to the similar poles of the other magnet, the other magnet should move away as if repelled by magnetic forces. If, on the other hand, opposite poles come close to one another, the free magnet should move closer and become stuck to the magnet being dragged.

To make things a bit more interesting one should be allowed to flip a magnet from end to end (swapping the poles) by clicking on the magnet without moving the mouse. This will provide a way to separate the two magnets if they get stuck together (since as soon as one of them is reversed it will repel the other).

To decide whether two magnets interact (attract or repel), we need to look at the distance between their opposite poles. So it is the poles rather than the magnets that really matter when deciding whether something should be attracted or repelled. As a result, instead of just manipulating magnet objects in your program, you will also need objects that explicitly represent poles.

Questions: How do you approach this problem? How do you decide how many classes to write? How do you decide what each class should do?

You could write the whole program in one class. But that is bad style. Not to mention cumbersome. To solve the problem elegantly you'd write at least two classes: one to handle magnets, one to put them on the window and handle the mouse.

A magnet has geometry (location, width, height), two poles, can move, and interact with the other magnet.

The MagnetController has to create two magnets, handle mouse movement by calling the appropriate magnet methods, and handle interaction (magnets do not know about each other).

Before looking at the solution, try to come up with an outline of the classes.

Note that the way you design your solution depends on a few decisions you need to take:

(1) How do you handle rendering of the magnets?

(2) How do you handle the interaction between magnets?

Reading

—code for Magnets