GridWorld
Activity 5 - Roadrunner 1

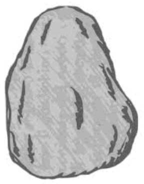
Objective: To create an exploding Boulder and Wile E Coyote simulation in GridWorld.

Background:
Wile E Coyote is a cartoon coyote who likes to chase a Roadrunner, another cartoon character, and put bombs in its path. You will create a **Coyote** critter to represent Wile E Coyote. You will also create three actors. The first is a **Stone** which is like a **Rock**, but the **Stone** can change into a **Boulder**. The **Boulder** is similar to a **Rock**, but it explodes after some time. A **Kaboom** is an actor that looks like an explosion and it shows up when the **Boulder** explodes. A **Kaboom** lasts only a few steps before it disappears. If the **Coyote** walks into a **Boulder**, the **Boulder** explodes and the **Coyote** is removed from the grid.

Assignment:

1. Create a **Kaboom** class that extends the **Actor** class. The **Kaboom** has two fields. One is the lifetime of the **Kaboom**, an integer number representing the number of steps (calls to act()) before it disappears. The other is an integer threshold constant (**final int**) that should be set to **3**. In the constructor, set the color to null and initialize the lifetime to the threshold number. The **Kaboom act()** counts down its lifetime, then removes itself from the grid. Here is the graphic of a **Kaboom**:

2. Create a **Boulder** class that extends the **Actor** class. The **Boulder** has the same two fields as **Kaboom**, but the threshold constant means something different. It is used to determine when the **Boulder** turns **red** just before it explodes. In the constructor, set the color to null and initialize the lifetime to a random number between 1 and 200 inclusive. A second constructor, similar to the first, passes an integer number that sets the lifetime. The **Boulder act()** counts down its lifetime. When the lifetime is less than the threshold, set the color to **red**. When the lifetime is equal to 0, replace the **Boulder** with a **Kaboom**. Here is the graphic of a Boulder:



3. Create a **Stone** class that extends the **Rock** class. The **Stone** has the same two fields as **Boulder**, but the threshold constant is used to determine when the **Stone** turns **green**. Green means it is about to turn into a **Boulder**. In the constructor, set the color to null and initialize the lifetime to a random number between 1 and 200 inclusive. A second constructor is similar to the first but passes an integer number that sets the lifetime. The **Stone act()** counts down its lifetime.

When the lifetime is less than the threshold, set the color to **green**. When the lifetime is equal to 0, replace the **Stone** with a **Boulder**. The **Stone** will look just like a **Rock**.

4. Test your classes. Use the **StoneRunner** class provided. It places **Stones** on a 20 by 20 grid. Run the simulation and watch to see each **Stone** turn green, then be replaced by a **Boulder**. When the **Boulder** turns red and some steps elapse, it is replaced by a **Kaboom** which disappears after a couple more steps.

5. Create a **Coyote** class (Wile E Coyote) that extends the **Critter** class. The **Coyote** drops **Stones** as he wanders around the grid. He moves in straight lines, but stops every couple steps to drop a stone and change direction. Here is what the **Coyote** looks like:

The **Coyote** works in this way:

a. When a **Coyote** is created, it has a color of null and its current direction is one randomly selected from 8 compass directions in 45˚ increments from 0 to 315.

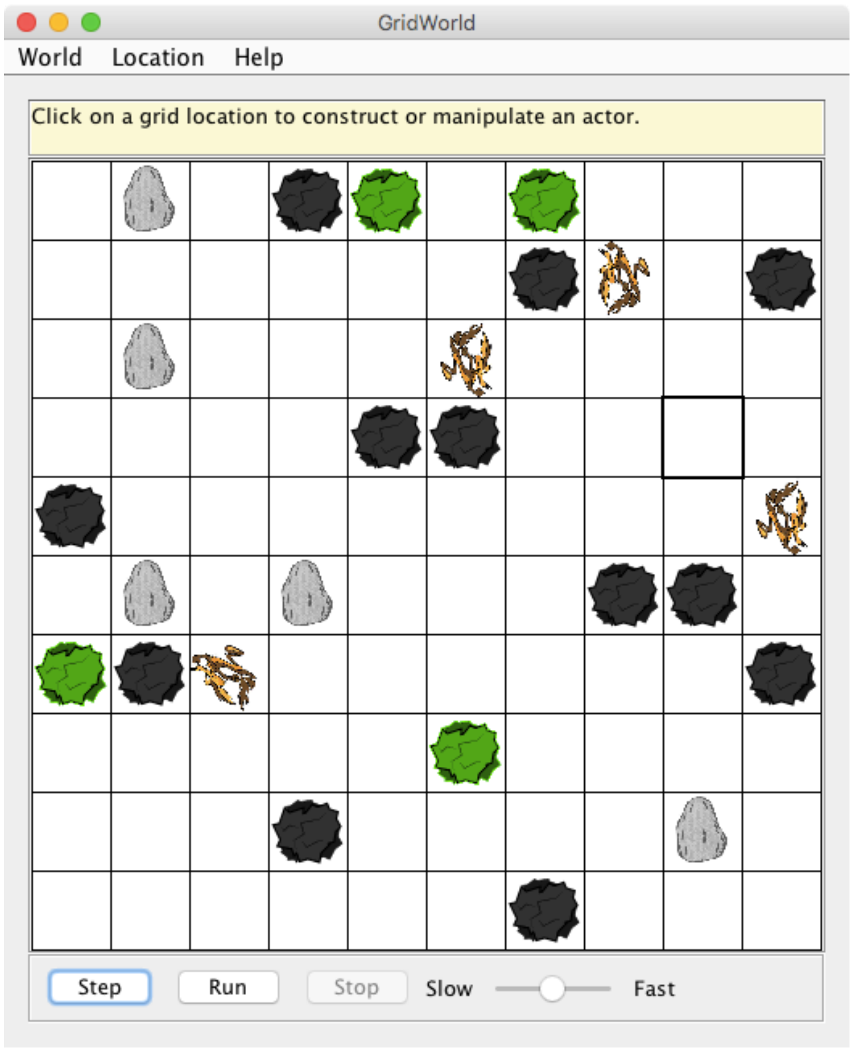
b. The **Coyote** walks in his current direction one cell at a time until he meets another **Actor** or bumps into the edge of the grid or walks 5 cells in a row, whichever comes first. If he walks into a **Boulder**, the **Boulder** explodes (replaced by a **Kaboom**), and the **Coyote** is removed from the grid.

c. At the end of part b (if the **Coyote** is still on the grid), the **Coyote** waits for 5 steps (sleeps). If he had walked into the wall, he picks a new direction at random, then repeats part b above. If he had not walked into a wall, he puts a **Stone** in an adjacent open cell, picks another direction at random, and repeats part b. (This prevents the **Coyote** from boxing himself in with **Stones** and the edge of the grid.)

A note about removing a **Critter** from the grid. The **removeSelfFromGrid** method will only work if it is the last executed method in the **makeMove** method. The safest thing to do is to put a **return** statement immediately following **removeSelfFromGrid** in the **makeMove** method to insure no other statements are executed.

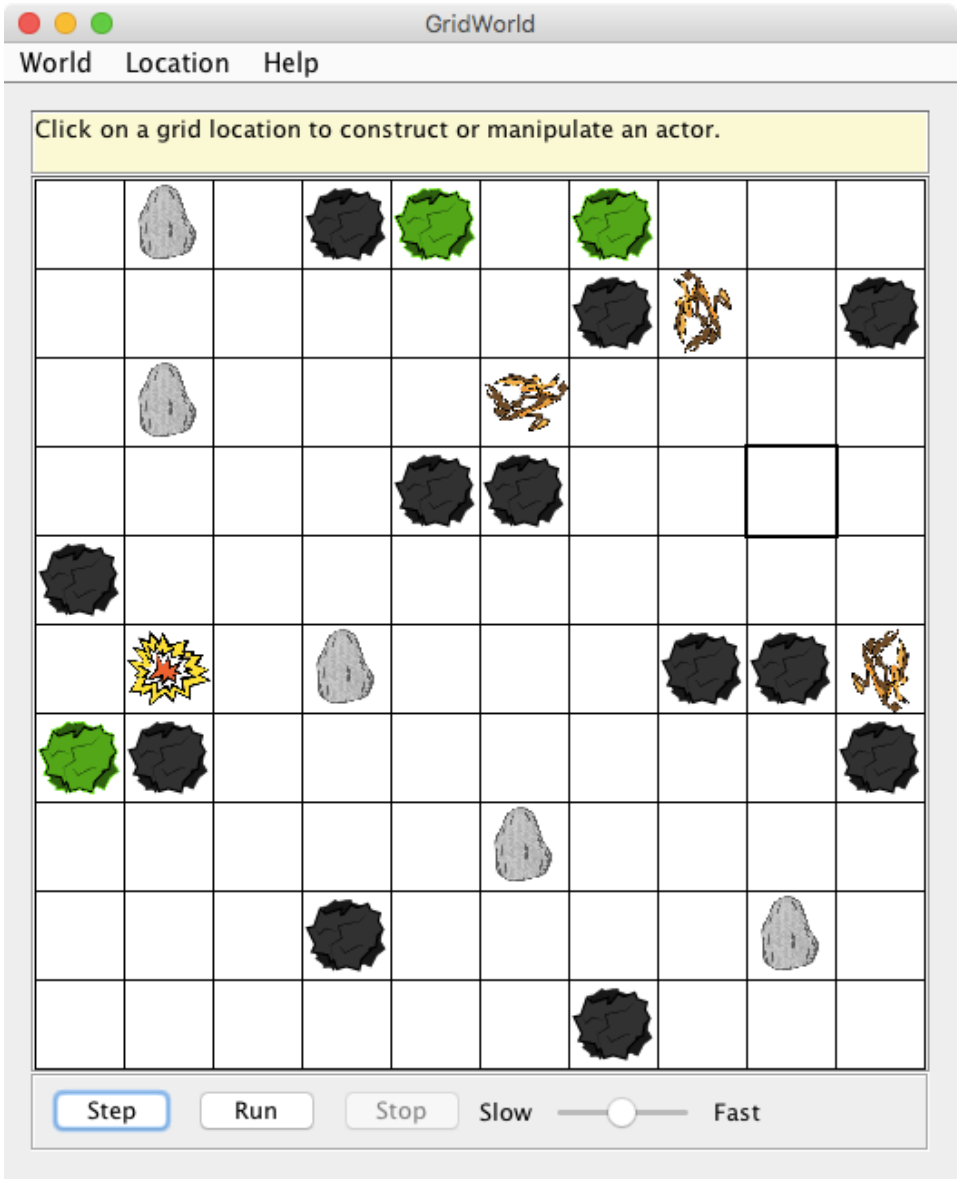
6. Create a **CoyoteRunner** that places 2 **Coyotes** on a 10 by 10 grid. Run the simulation. Watch to see each that the **Coyote** walks, waits, places **Stones**, and explodes when it walks into **Boulders**.

Here is a screen shot of **Coyotes**, **Stones** (black or green), and **Boulders** (gray or red). One **Coyote** is about to walk into a **Boulder** (lower left).



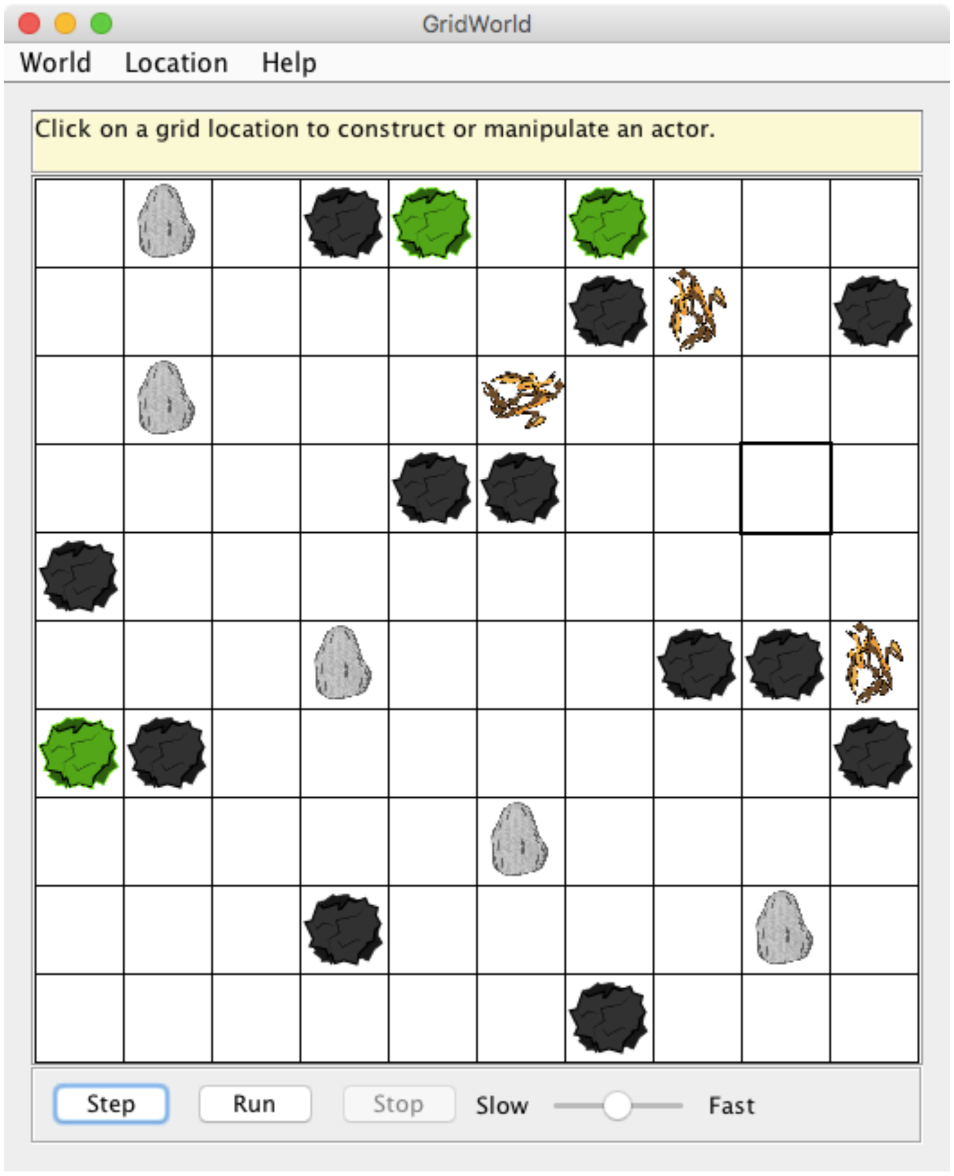
Click on a grid location to construct or manipulate an actor.

Here is a screen shot after the **Coyote** walks into the **Boulder**.



Click on a grid location to construct or manipulate an actor.

Here is a screen shot a few steps later.



Click on a grid location to construct or manipulate an actor.