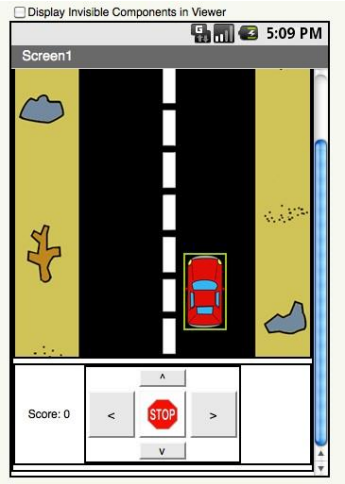
Lesson 5 (1 - 2 days)

Lesson Overview
In this lesson students will develop a driving game using the skills they learned in the previous lesson.

Objectives
Students will refine their skills with the clock, button, and acceleramotor methods developed in the last lesson

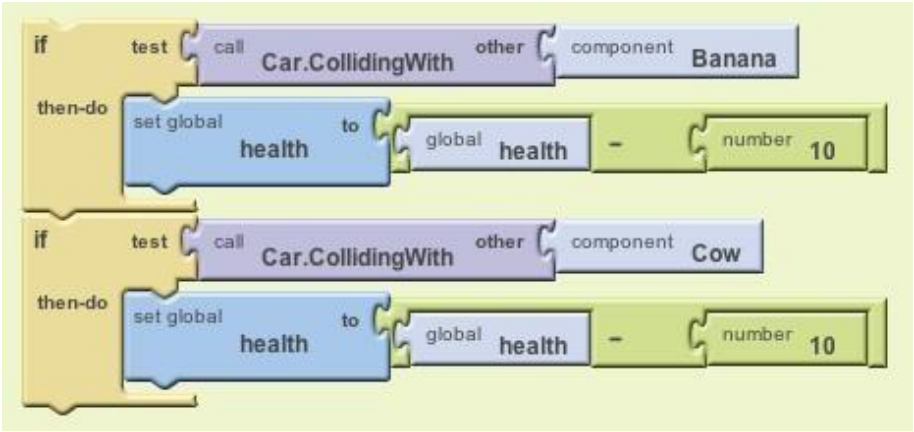
Activity
Review previous day's material -- Canvas, Drawing, and motion of sprites -- by building a simple game where a car (represented by an ImageSprite) moves left and right on a road (a Canvas with an image as a background):



The students may implement the motion of the car using either the Clock component, buttons, or the accelerometer. If you want to extend this project, make them choose two methods in which to utilize.

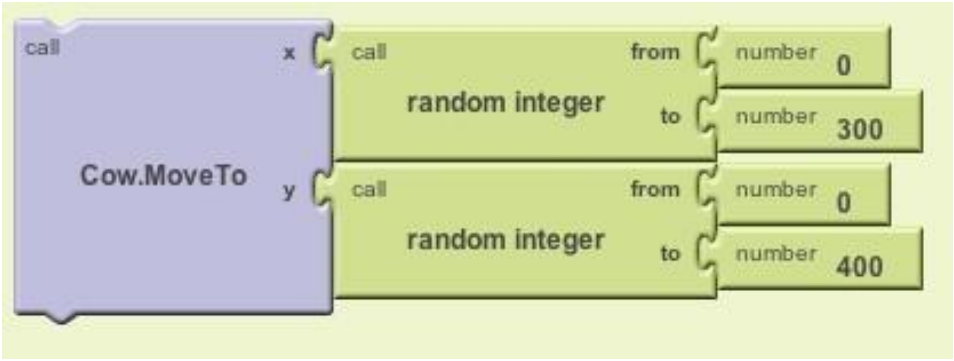
Using this same project as a starting point, introduce students to **collision detection**, a very useful facet of game development. This allows their sprites to interact with other sprites on the screen. Ask them to add one or more obstacles (ImageSprite objects) onto the road, e.g. the banana and the cow in the following image:

In the blocks code, ask students to insert if-statements (right after the code that updates the car's position) to check if the car is colliding with other obstacles:



Next, introduce students to **random-number generation** by showing them how to make the obstacles on the road appear in random (x,y) coordinate locations using the *random integer* procedure to pick random values for x and y.

In the below blocks code, the Cow's x value is picked randomly from within the range [0, 300] and the y value is picked randomly from the range [0, 400]. This places it randomly in between those coordinates on the screen.



Extra Credit: If students finish early, have them attempt to complete this car obstacle---avoidance game by animating the obstacles such that they appear at the top of the screen, move from top to bottom, and then reset to the top of the screen (at a random x location) and come down again. The students can accomplish this either manually using Clock components or automatically using the built---in ImageSprite properties of Interval, Speed, and Heading (the heading would be 270 degrees if the objects are moving straight down). Note: If multiple obstacles are to be moved simultaneously but at different rates, one Clock component might be necessary for each obstacle.