Homework 11

The InClass is to be demonstrated (or mailed) to your group leader. To make best use of the time you have in labs, be sure you have
(1) read the assignment before lab
(2) read appropriate background material before lab

The Homework is to be submitted via Eagle. It is to be individual work. You may talk to other students in the course about your design and for ideas, but you are to write the complete Alice programs by yourself. You may receive help from the Professor, CS Tutors, UTF’s or TA. In your comments, identify (by name) all those who helped you. Failure to do so is considered cheating.

In most cases, it will not be possible to finish the homework during labs. Plan on spending several hours per week outside of class to get the homework done.

Inclass (5 points) (Show to group leader during lab Wednesday or Friday.)
Consider the set up below. Do the following:

- (1 point) Create three moles (Amusement Park) and one bopper (amusement park)
- (1 points) Generate a variable to store "which mole" should be bopped.
- (3 points) Generate a random number between 1 and 3. If the number is one, bop the leftmost mole. If the number is two, bop the middle mole, otherwise bop the rightmost mole. The a mole is "bopped": the bopper moves to the selected mole and the other two moles disappear. Try this several times or put the code in a loop so the program itself does the re-trying. We should see that a different mole is selected on the different trials.
- (1 point BONUS) use a nested if to accomplish the bopping

Hints:
- " Random number" is in the world/functions tab.
- For random numbers, if you pick 1 as the minimum value and 4 as the maximum, you will get values of 1,2 or 3 - provided you ask for "integer" rather than fractional results. (In other words, the random number generated will be up to the maximum number, but not including it.)
- If you want to "see" the random number that is generated, you can drag the "print" statement (from the bottom of the menu) to the screen.
• You MAY (inadvertently) be exposed to "bugs" in a language. When you generate a random number and compare to an integer, ON SOME COMPILERS it always returns a false. It is because they are storing the constant as 1.0 instead of 1. (In computer language, we say one is stored as a double and the other is stored as an integer. Different representations can cause problems in trying to compare for equality.)

Alice is a bit FLAKY - as it isn't a production language (used in the business world) - just a teaching tool. A major goal of Alice is to be able to visualize what the commands do. At that it is successful. It doesn't handle types correctly, and there are numerous other glitches you will discover.

ANYWAY, there are two ways around the problem of being unable to correctly compare for equality:

1. randVar set value to randVar*1  //the multiplication changes the type of randVar
2. compare randVar <2 instead of randVar==1

Homework (due Friday at midnight)

Create a simulation of a bumper car ride using two instances of Wheely (Web/Fantasy) and a platform (Web/Fantasy) where the two wheelies move continuously around within the arena. (The Amusement park gallery actually has a bumper car class, but it was harder to see what was going on.) In this animation, each wheely should be moving forward a small amount until it gets too close to the other wheely or to the edge. Set the duration to a small increment of time so it moves faster. When the wheely gets too close to an edge - then turn the wheely 0.2 of a revolution clockwise (to get a different direction) and continue moving forward. When the wheelies get too close (I used two meters), make a "thud" sound, have the two wheelies both “move to” the camera, and then quit the animation.

Note, this needs to work no matter WHERE on the platform the wheelies are initially placed. It is not sufficient to "hard code" a set of motions that only work from a specific starting point. The animation needs to work regardless of initial placement. Thus, we should see a pattern of repeated motion and testing for conditions.

Point Distribution
1. (2 points) Setup
2. (5 points) Two wheelies moving forward in a while loop
3. (5 points) Wheelies turning near the edge based on an if condition
4. (2 points) Crashing
5. (1 point) Actually stopping the animation when the crash occurs
6. (1 BONUS point ) Use a boolean variable "crash" and set it to true when there is a crash.

Hints:
• I used the “Speed:1x” slider bar in the play environment so the animation moves faster.
• To avoid a wheely driving off the edge, a simple form of collision detection is needed. One way to check for a possible collision is to use the “distance to” function to compute the distance of the wheely to the edge. Remember that “distance to” is measured “center to center”. In this world, a measurement from the wheely to the center of the platform is exactly what you need. When it gets too far from the center, it will fall off the edge. I used a distance of greater than \( \frac{1}{2} \) of the width of the platform as the point in which the wheely turned.

• Pseudocode is really helpful in getting the overall plan. The "very rough" pseudo code might look like:

```plaintext
while wheelies have not crashed
    move each wheely forward a bit
    if either wheely needs to turn, it should turn
    if the two wheelies are too close, note that they have crashed

Do the actions associated with the crash
```